MPRA
Munich Personal RePEc Archive

An Adaptive Succesive Over-relaxation Method for Computing the Black-Scholes Implied Volatility

Li, Minqiang

Gerogia Institute of Technology

21 January 2008

Online at https://mpra.ub.uni-muenchen.de/6867/
MPRA Paper No. 6867, posted 24 Jan 2008 05:37 UTC

An Adaptive Successive Over-relaxation Method for Computing the Black-Scholes Implied Volatility

January 21, 2008

A new successive over-relaxation method to compute the Black-Scholes implied volatility is introduced. Properties of the new method are fully analyzed, including global well-definedness, local convergence, as well as global convergence. Quadratic order of convergence is achieved by either a dynamic relaxation or transformation of sequence technique. The method is further enhanced by introducing a rational approximation on initial values. Numerical implementation shows that uniformly in a very large domain, the new method converges to the true implied volatility with very few iterations. Overall, the new method achieves a very good combination of efficiency, accuracy and robustness.

Keywords: Successive over-relaxation, Black-Scholes formula, Implied volatility, Rational approximation.

JEL Classifications: C00, G13

1. Introduction

A breakthrough in modern financial theory is the Black-Scholes-Merton theory of option pricing, developed by Black and Scholes (1973), Merton (1973, 1976) and many others. In 1997, the Nobel prize in Economics was awarded to Merton and Scholes for the discovery and extension of this theory. As the press releases described it, the Black-Scholes theory is "among the foremost contributions to economic sciences over the last 25 years." The ultimate result of the theory is the celebrated Black-Scholes formula, which is being used daily by traders around the world nowadays. Besides being used to price options on stocks, the Black-Scholes theory is also used to value options on futures, options on currencies, options on interest rates, etc. Even models that are out of the Black-Scholes framework, such as stochastic volatility models and more recently Levy-process type models, still refer to the Black-Scholes model as the basic benchmark. See, for example, Heston (1993), and Carr and Wu (2003). The use of the Black-Scholes formula is so pervasive that Nigel Goldenfeld, a Physics professor at the University of Illinois at Urbana-Champaign, claims that the Black-Scholes formula is *the* most frequently used equation by human beings nowadays, beating both Newton's laws of motion in classical mechanics and Schrödinger's equation in quantum mechanics.

In practice, traders often do not use the Black-Scholes formula to price options. Rather, they observe the actual price on the market and then use the Black-Scholes formula backwards to compute the volatility parameter, called the implied volatility. Early academic development of this concept and on its applications includes Latané and Rendleman (1976), and Schmalensee and Trippi (1978). The implied volatility is a useful quantity because it is a succinct way to talk about option prices. In fact, traders are usually more comfortable to talk in terms of implied volatilities than prices themselves. Market participants also pay close attention to the implied volatility since it serves as a forward-looking measure of people's expectation about future market movements. The Black-Scholes volatility is also a useful concept for models outside the Black-Scholes framework. For example, people are constantly interested in whether a new proposed model can produce the Black-Scholes implied volatility curve observed on the market. The importance of the implied volatility is most evident in the classical NatWest Markets case. In 1997, NatWest lost 90.5 million pounds due to consistent underestimating of implied volatilities it used to price interest rate swaptions.

As early researchers have noted in their works, the inversion of the Black-Scholes formula cannot be done in closed form if elementary functions are to be used. Thus, to get the implied volatility, people usually use some solver method, for example, the Dekker-Brent algorithm, or the Newton-Raphson algorithm. Both methods have strengths and weaknesses (see Jäckel 2006, and Li 2008). The Newton-Raphson algorithm has a quadratic convergence order and is thus quite efficient. However, it can suffer from numerical instabilities. For example, each step in the Newton-Raphson algorithm requires the division by the option vega, which can be extremely small for away-from-the-money options. Also, with bad starting values, the Newton-Raphson algorithm might not converge at all to the true implied volatility. The Dekker-Brent algorithm uses a combination of bisection, secant and inverse quadratic interpolation, and is guaranteed to converge. Some commercial software, such as MATLAB, chooses to use the Dekker-Brent algorithm, possibly because of its robustness. However, the Dekker-Brent algorithm is usually slower than the Newton-Raphson algorithm. For example, as Li (2008) reports, the number of inversions one has to do for the S&P 500 index options for the period January 1996 to December 2004 is about 700,000. On a typical computer, to achieve 10^{-6} accuracy in total volatility (defined as the volatility multiplied by the square root of maturity), it takes the built-in function "blsimpv" in MATLAB hours to finish these inversions, while the Newton-Raphson algorithm without vectorization takes about 960 seconds.

Many research works have focused on inverting the Black-Scholes formula. Manaster and Koehler (1982) show that if one starts the initial estimate for the Newton-Raphson algorithm from the inflection point, then in theory the algorithm always converges. Jäckel (2006) analyzes the Newton-Raphson algorithm in detail and points out various pitfalls one should avoid. He also points out how one can avoid those pitfalls by designing a series of precautionary measures. Researchers have also considered alternatives to the solver methods, such as analytical approximation or quasi-iterative methods. Brenner and Subrahmanyam (1988) develop a simple formula for computing the implied volatility of at-the-money options. Bharadia et al. (1996) derive a simple volatility formula which does not require the option to be exactly at-the-money. Chance (1993, 1996) develops a formula for computing the implied volatility by adding a quadratic adjustment term to the Brenner-Subrahmanyam formula. Chambers and Nawalkha (2001) develop a simplified extension of Chance's method and obtain a formula that is more accurate than previous methods. One shortcoming of the Chance method and the Chambers-Nawalkha method is that they both require the knowledge of the at-the-money implied volatility. Based on the linearization of the cumulative normal distribution, Corrado and Miller (1996a, 1996b) develop a simple and elegant formula that overcomes this shortcoming. This formula is quite accurate for near-the-money options but unfortunately its performance deteriorates when the option gets away from the money. Kelly (2006) considers inverting the Black-Scholes formula using the implicit function theorem. He also studies the numerical errors of various methods. Charogy-Corona and Ibarra-Valdez (2006) consider an approximation formula for the implied volatility by means of an asymptotic representation of the Black-Scholes formula. However, they do not give a numerical analysis. Li (2008) takes a new approach by using a rational approximation for the implied volatility. Since a rational approximation can have a small uniform error in a large domain, the approximation domain considered in Li (2008) is fairly large, with moneyness ranging from -0.5 to 0.5 and total volatility ranging from 0 to 1. For most index options or currency options, this domain is large enough. It is found that the rational approximation is the most accurate among all existing closed-form methods, achieving 10^{-3} in terms of uniform error in the total volatility. Also, the rational approximation is able to invert one million options within a few seconds.

In this paper, we introduce a new method to compute the Black-Scholes implied volatility based on successive over-relaxation. Thus, our method is an iterative method and not in closed form. We study this method for two reasons. First, although the approximation domain in Li (2008) is fairly large, in practice, total volatility can be well above 1, especially for energy derivatives, or equity derivatives near earnings announcements. Thus, we use a much larger domain in this paper, with moneyness ranging from -3 to 3 and total volatility ranging from 0 to 6. This domain should cover almost all practical applications. Second, we try to search for a method which has the best combination of efficiency, accuracy and robustness. We will see that our method can achieve quadratic order of convergence, the same order as the Newton-Raphson algorithm. On the other hand, our method is robust as is the case for the Dekker-Brent algorithm. With just five iterations our method achieves a uniform error in the total volatility in the order of 10^{-13} inside the huge domain. The time it takes to invert one million options is about 10 seconds.

Our first contribution in this paper is that we design a new method to compute the implied volatility. The basic idea of our method is to split the Black-Scholes formula into two parts and to search for the implied volatility iteratively. To improve efficiency, we introduce a relaxation parameter. We call our method with over-relaxation the `SOR` algorithm. Unlike the Newton-Raphson algorithm, the computation of vega is not needed in our method. We analyze the well-definedness, local and global convergence properties of the `SOR` algorithm in detail. In Theorem 3.1, we show that when the relaxation parameter is large enough, the `SOR` algorithm is always globally well-defined. This is a very attractive feature because it gives rise to robustness of our algorithm. Theorem 3.2 shows that the local convergence is dictated by the behavior of the iteration function near the true implied volatility. Theorem 3.3 shows that only two convergence patterns can occur, monotone or oscillating. Theorem 3.4 and 3.5 show the convergence patterns of price errors and implied volatility errors, respectively. The most interesting analysis is on global convergence, which is usually very difficult to study. We obtain many interesting results. Theorem 3.6 and 3.7 study two important special cases where the moneyness or the relaxation parameter is 0, respectively. Theorem 3.8 is shows that when the relaxation parameter is larger than some threshold value, the `SOR` algorithm is always globally convergent. The case where the relaxation parameter is smaller than the threshold value is much harder to analyze and we content ourselves by giving three sufficient conditions on global convergence in Theorem 3.9, 3.10 and 3.11.

The second contribution of this paper is that we introduce two convergence acceleration techniques to the `SOR` algorithm. The basic `SOR` algorithm usually has a linear order of convergence, as is shown in Theorem 4.1. The first acceleration technique is based on dynamic relaxation, where we adaptively adjust the relaxation parameter at each iteration. We call this the `SOR-DR` algorithm. Theorem 4.2 shows the global well-definedness property and convergence patterns for the `SOR-DR` algorithm, while Theorem 4.3 shows that it has a quadratic order of convergence. The second acceleration technique is based on a time-honored sequence transformation technique, where at each iteration step we perform a nonlinear extrapolation, with the weight adaptively adjusted. We call this the `SOR-TS` algorithm. Theorem 4.4 shows the global well-definedness and convergence properties and local convergence patterns for the `SOR-TS` algorithm, while Theorem 4.5 shows that it has a quadratic order of convergence.

Our third contribution is on numerical implementation of our algorithms. To further enhance the efficiency of our algorithms, we introduce a rational approximation on the initial estimates, in the same spirit as Li (2008). We show numerically that uniformly in a very large domain, our accelerated algorithms converge to the true implied volatility with very few iterations. We also briefly compare the `SOR-TS` algorithm with other methods such as the Newton-Raphson algorithm and the Dekker-Brent algorithm, and show that the `SOR-TS` algorithm achieves the best combination of accuracy, efficiency and robustness.

Finally, we extend our successive over-relaxation method to the computation of implied correlation in the Margrabe formula, the normal implied volatility in the Bachelier formula, and the critical stock price in a compound call on call option. The extension to the Margrabe formula is straightforward. The extensions to the Bachelier implied volatility and critical stock price require some analysis, and Theorem 6.1 and 6.2 show that for both applications, the successive over-relaxation method is globally well-defined and convergent with a quadratic order of convergence. These three examples demonstrate that the idea of successive over-relaxation is not limited to the computation of the Black-Scholes implied volatility, but rather applicable in a much wider range of financial problems.

The rest of the paper is organized as follows. Section 2 introduces the problem of computing the Black-Scholes implied volatility. Section 3 introduces the basic successive over-relaxation method and studies its well-definedness, local and global convergence properties. Section 4 considers two acceleration techniques, one based on dynamic relaxation and the other on sequence transformation, and shows that both of them achieve quadratic order of convergence. Section 5 implements all three algorithms with a rational approximation enhancement. Section 6 demonstrates that the successive over-relaxation method is applicable in many other financial problems through three additional examples. Section 7 concludes. Proofs are in the Appendix.

2. The Implied Volatility Problem

Let C be the price of a call option at time t with expiration date T and strike price K. Let S be the current stock price, σ the volatility of the stock, r the risk-free interest rate, and δ the dividend rate. Then the Black-Scholes formula expresses C in closed form as follows:

$$C(S,t;r,\sigma,T,K,\delta) = Se^{-\delta(T-t)}N(d_1) - Ke^{-r(T-t)}N(d_2),$$

where

$$d_1 = \frac{\log(Se^{(r-\delta)(T-t)}/K)}{\sigma\sqrt{T-t}} + \frac{1}{2}\sigma\sqrt{T-t}, \quad d_2 = d_1 - \sigma\sqrt{T-t},$$

and $N(\cdot)$ is the cumulative normal distribution function.

Let us first define the normalized call option price $c(\cdot,\cdot)$ by

$$C(S,t;r,\sigma,T,K,\delta) = Se^{-\delta(T-t)}c\big(\log(Se^{(r-\delta)(T-t)}/K), \sigma\sqrt{T-t}\big).$$

The function $c(x,v)$ is given by

$$c(x,v) = N\Big(\frac{x}{v}+\frac{v}{2}\Big) - e^{-x}N\Big(\frac{x}{v}-\frac{v}{2}\Big), \tag{1}$$

where the moneyness x and total volatility v are given by

$$x = \log(Se^{(r-\delta)(T-t)}/K), \quad v = \sigma\sqrt{T-t}. \tag{2}$$

A call option with $x > 0$, $x = 0$ and $x < 0$ is said to be in-the-money, at-the-money, and out-of-the-money, respectively. Following Li (2008), we call equation (1) the dimensionless Black-Scholes formula.

The dimensionless Black-Scholes formula implicitly gives v as a function of c and x and is the starting point for our successive over-relaxation method. This dimensionless formula clearly states that the Black-Scholes formula is essentially a relation between three dimensionless quantities, namely the normalized price c, the integrated volatility v and the moneyness x. Given observed values of S, t, r, T, K, δ and option price C, we first calculate c according to $c = C/(Se^{-\delta(T-t)})$ and x according to equation (2), and then compute v through some algorithm. The implied volatility σ can then be obtained by dividing v by $\sqrt{T-t}$.

Figure 1 gives a surface plot for the normalized call option price c when the moneyness x ranges from -3 to 3, and the volatility v ranges from 0 to 6. As we see, when $|x|/v$ is large, c is very insensitive to v and the inversion is not very meaningful because a tiny change in $c(x,v)$ due to measurement error can give rise to a huge change in v. Thus we will omit these regions in the numerical implementation of our algorithms in Section 5 by imposing the restriction $|x|/v \le 3$.

We only need to consider inverting call options because by the put-call parity in the Black-Scholes theory (see Stoll 1969), we have $C = P + Se^{-\delta(T-t)} - Ke^{-r(T-t)}$. Writing out in normalized call and put prices, we have

$$c(x,v) = p(x,v) + 1 - e^{-x}, \tag{3}$$

where the normalized put price is defined as the ratio of the actual put price P and the quantity $Se^{-\delta(T-t)}$. If we are given a put option with moneyness x and normalized put price p, we will simply invert a call option with moneyness x and normalized call price $c = p + 1 - e^{-x}$.

Finally, we only need to consider options with moneyness $x \le 0$. This is because we have the following symmetry, which we call the "in-out" duality:

$$c(-x,v) = e^{x}c(x,v) + 1 - e^{x}. \tag{4}$$

If we need to invert the Black-Scholes formula for an option with moneyness $x \geq 0$ and normalized call price c, we can as well use the right hand side of the above equation to invert its dual option with moneyness $x' = -x \leq 0$ and normalized call price $c' = e^x c + 1 - e^x$. This is also in line with industry practice of treating out-of-the-money options as more informative than in-the-money options. Another rationale to invert out-of-the-money options is the following. Letting $\widehat{v}$ be the estimate of the true volatility v_* that one gets from inverting the dual out-of-the-money option when $x > 0$, then the "in-out" duality gives

$$|c(-x, \widehat{v}) - c(-x, v_*)| = e^x |c(x, \widehat{v}) - c(x, v_*)| > |c(x, \widehat{v}) - c(x, v_*)|.$$

That is, if we use "in-out" duality to compute the implied volatility for an in-the-money option, the error in terms of price for the in-the-money option is always smaller than that for its dual out-of-the-money options.

3. The Successive Over-relaxation Method

3.1. The SOR algorithm

In this section, we establish a numerical method to compute the implied volatility which is based on the idea of successive over-relaxation. We also analyze the properties of the new method, including global well-definedness, local convergence, as well as global convergence.

We will use $n(\cdot)$ to denote the standard normal density function. The following functions will play important roles, so we give them specific names in order to shorten expressions:

$$n^+(x, v) = n(x/v + v/2), \tag{5}$$

$$N^+(x, v) = N(x/v + v/2), \tag{6}$$

$$N^-(x, v) = e^{-x} N(x/v - v/2), \tag{7}$$

$$c^+(x, v) = N^+(x, v) + N^-(x, v). \tag{8}$$

Notice that the dimensionless Black-Scholes formula now becomes $c(x, v) = N^+(x, v) - N^-(x, v)$. By the discussion in Section 2, we only consider $x \leq 0$ and $v > 0$. The following lemma is very useful later when we analyze the properties of our method. Because of the well-known discontinuity of the Black-Scholes formula, we consider the $x < 0$ and $x = 0$ cases separately.

Lemma 3.1. *When $x < 0$, we have*

1) *$N^+(x, v)$ is strictly increasing in v. Furthermore, $N^+(x, 0^+) = 0$ and $N^+(x, +\infty) = 1$;*
2) *$N^-(x, v)$ is strictly increasing in v for $v \leq \sqrt{2|x|}$ and decreasing in v for $v > \sqrt{2|x|}$. Furthermore, $N^-(x, 0^+) = N^-(x, +\infty) = 0$;*
3) *$c(x, v)$ is strictly increasing in v, strictly convex in v for $v < \sqrt{2|x|}$ and strictly concave in v for $v > \sqrt{2|x|}$. Furthermore, $c(x, 0^+) = 0$ and $c(x, +\infty) = 1$;*

4) $c^+(x,v)$ is strictly increasing in v. Furthermore, $c^+(x,0^+)=0$, and $c^+(x,+\infty)=1$;

5) Let $v>0$, $u>0$ and $v\neq u$. Then

$$\frac{u^2}{2\,|x|}\left[c^+(x,v)-c^+(x,u)\right] < c(x,v)-c(x,u) < \frac{v^2}{2\,|x|}\left[c^+(x,v)-c^+(x,u)\right]. \tag{9}$$

When $x=0$, we have

1) $N^+(0,v)$ is strictly increasing in v. Furthermore, $N^+(0,0^+)=1/2$ and $N^+(0,+\infty)=1$;

2) $N^-(0,v)$ is strictly decreasing in v. Furthermore, $N^-(0,0^+)=1/2$ and $N^-(0,+\infty)=0$;

3) $c(0,v)$ is strictly increasing in v, and strictly concave in v. Furthermore, $c(0,0^+)=0$ and $c(0,+\infty)=1$;

4) $c^+(0,v)\equiv 1$.

Hereafter, we will suppress the dependence on x in the functions n^+, N^+, N^-, c, and c^+ unless there can be a confusion. That is, we will write $N^+(v)$ for $N^+(x,v)$, etc. To introduce our successive over-relaxation method, we need to define the iteration function. Let $N^{-1}(\cdot)$ denote the inverse function of the cumulative normal distribution $N(\cdot)$. Let the moneyness be x, the observed option price c_*, and the true implied volatility v_*. That is, $c_*=c(x,v_*)$. For fixed $c_*=c(x,v_*)$ and x, define a function $F(v;x,v_*,\omega)$ as follows

$$F(v;x,v_*,\omega)\equiv c_*+N^-(v)+\omega N^+(v). \tag{10}$$

The parameter ω will play the role of the relaxation parameter in our method. The iteration function is given by

$$G(v;x,v_*,\omega)=N^{-1}\left(\frac{F(v;x,v_*,\omega)}{1+\omega}\right)+\sqrt{\left[N^{-1}\left(\frac{F(v;x,v_*,\omega)}{1+\omega}\right)\right]^2+2|x|}. \tag{11}$$

Notice that although we write G and F as functions of v_*, they only depend on v_* through $c_*=c(x,v_*)$. In order for $G(v;x,v_*,\omega)$ to be well-defined, we will require $\omega>-1$ throughout the paper. We will write $G(v;x,v_*,\omega)$ as $G(v)$ unless we want to emphasize the dependence of G on x or ω. Similarly for $F(v;x,v_*,\omega)$. Our problem is to compute the v_* from the observed c_* and x. For a given sequence v_k of implied volatility estimates, we will write c_k for the sequence $c(x,v_k)$. Our method of finding v_* is the following:

```
1.  Select an initial point $v_0$ and a fixed relaxation parameter $\omega$;
2.  (SOR) After obtaining $v_k$, compute $v_{k+1}$ from the following equation
```

$$v_{k+1}=G(v_k); \tag{12}$$

```
3.  Stop when $|v_k - v_*| < \epsilon$ or $|c_k - c_*| < \epsilon$, for some $\epsilon$ small.
```

We will call the above algorithm the `SOR` algorithm. This algorithm is an interesting application of the successive over-relaxation idea to financial problems where the underlying relations are nonlinear. The successive over-relaxation method was first developed by Young and others around 1950. The most well-known application of successive over-relaxation is to solve systems of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$. In finance, the `PSOR` (projected successive over-relaxation) algorithm is widely used in solving partial differential equations for derivative prices. In Nash (1990), Young reviews the historical development of iterative methods. This book also contains an interesting account of how many iterations it took him to finally get his dissertation published in 1954 (Young 1954).

Some simple algebra shows that equation (12) is equivalent to

$$c_* + N^-(x, v_k) + \omega N^+(x, v_k) = (1+\omega)N^+(x, v_{k+1}), \quad k = 0, 1, \cdots. \tag{13}$$

By Lemma 3.1, $c(x, v)$ is strictly increasing in v, hence v_* is the unique fixed point of $G(v)$. This in turn implies that if the `SOR` algorithm converges, it will converge to v_*.

It is helpful to understand the `SOR` algorithm without relaxation. When ω is set to be 0, equation (13) reduces to

$$c_* + N^-(x, v_k) = N^+(x, v_{k+1}), \quad k = 0, 1, \cdots. \tag{14}$$

That is, we split the Black-Scholes formula into two parts and try to find a fixed point for the above iteration. The equation also shows the need to introduce the relaxation parameter. When x is very negative or v_* is very large, $N^-(x, v_k)$ can be very small so the convergence could be extremely slow. The relaxation parameter ω helps in these situations.

A well-known result is that if $G(v)$ is locally contracting (that is, locally Lipschitz around v_* with coefficient strictly less than 1), then for v_0 sufficiently close to v_*, the `SOR` algorithm will converge to v_*. See, for example, Isaacson and Keller (1994). In particular, the local Lipschitz condition can be quickly checked by the first-order derivative of $G(v)$ at v_*. Lemma 3.4 below gives the expression of $G'(v_*)$. If in addition, $G(v)$ is globally contracting (that is, globally Lipschitz with coefficient strictly less than 1), then starting from any v_0, the `SOR` algorithm converges to v_*.

Unfortunately, in general the function $G(v)$ does not have these global or even local Lipschitz properties. In fact, it is even possible for $G(v)$ to be not defined. From the definition of $G(v)$, this happens when the following condition is violated

$$0 < F(v; x, v_*, \omega) < 1 + \omega. \tag{15}$$

Because a thorough understanding of the function $G(v)$ is crucial, we plot the function $G(v)$ and its derivative in v in Figure 2 for five different parameter combinations (x, v_*, ω). For ease of exposition, we fix $x = -0.5$ and vary v_* and ω. Figure 2 shows that all of the following five behavior of $G(v)$ can occur, corresponding to the five rows of the subplots:

1. $G(v)$ **is globally well-defined and globally contracting.** This corresponds to the first row of Figure 2, where $v_* = 0.5$ and $\omega = 1$. This is the best case scenario where we can start from any positive v_0 and the `SOR` algorithm always converges.

2. $G(v)$ **is globally well-defined, not globally contracting, but locally contracting around** v_***.** This corresponds to the second row of Figure 2, where $v_* = 1.2$ and $\omega = 0$.
3. $G(v)$ **is globally well-defined, but not locally contracting around** v_***.** This corresponds to the third row of Figure 2, where $v_* = 1.8$, $\omega = -0.28$, and $G'(v_*) \doteq -1.32$.
4. $G(v)$ **is not globally well-defined, but locally contracting around** v_***.** This corresponds to the fourth row of Figure 2, where $v_* = 2.4$, $\omega = -0.1$, and $G'(v_*) \doteq -0.89$.
5. $G(v)$ **is not globally well-defined, and not locally contracting around** v_***.** This corresponds to the last row of Figure 2, where $v_* = 1.2$, $\omega = -0.66$, and $G'(v_*) \doteq -2.47$.

In the rest of this section, we will analyze the global well-definedness property, local convergence properties, as well as the global convergence properties of the `SOR` algorithm.

3.2. Global well-definedness property of the `SOR` algorithm

We will look at well-definedness first. It is extremely difficult to analyze local well-definedness for a fixed starting point v_0. The following numerical example illustrates this point. Let $x = -0.5$, $v_* = 2.5$ and $\omega = -0.1$. Then $G(v)$ is not defined for $v \in [0.548467, 1.46434]$. This means that we cannot start the `SOR` algorithm from a point in this region. However, if $v_0 = 0.03$, then the sequence jumps past this bad region to $v_1 = 2.2285$ and converges to v_*. Thus, in the following we look for a global well-definedness condition which guarantees that the sequence v_k is well-defined regardless of the initial point v_0. We will focus on $x < 0$ since we show later that the well-definedness property with $x = 0$ is quite simple to analyze even for a fixed starting point v_0.

First we define a function $\widetilde{v} = \widetilde{v}(w; x)$ as follows. For any $x \le 0$ and $-1 < \omega < 1$, we let

$$\widetilde{v} = \widetilde{v}(\omega; x) = \sqrt{2|x|}\sqrt{\frac{1+\omega}{1-\omega}}. \tag{16}$$

We will often simply write $\widetilde{v}$ or $\widetilde{v}(\omega)$ for $\widetilde{v}(\omega; x)$. We first establish two lemmas.

Lemma 3.2. *Let $x < 0$. Then, we have $\lim_{v \to 0^+} F(v) = c_*$ and $\lim_{v \to +\infty} F(v) = c_* + \omega$. Also, if $\omega \in (-1, 1)$, then $F(v)$ strictly increases on $(0, \widetilde{v})$ and strictly decreases on $(\widetilde{v}, \infty)$. If $\omega \ge 1$, then $F(v)$ strictly increases for all $v > 0$.*

Lemma 3.3. *Let $x < 0$. Suppose $\omega \in (-1, 1)$. Define*

$$H(\omega) = H(\omega; x, v_*) \equiv c_* + N^-(\widetilde{v}(\omega)) + \omega N^+(\widetilde{v}(\omega)) - (1 + \omega). \tag{17}$$

Then, $H(\omega)$ has a unique root $\widehat{\omega}$ in $(-1, 1)$. Furthermore, if $\omega \le \widehat{\omega}$, then $H(\omega) \ge 0$, and if $\omega > \widehat{\omega}$, then $H(\omega) < 0$.

Notice that $\widehat{\omega}$ is implicitly a function of x and v_*. For each fixed $\omega \in (-1, 1)$, the function $F(v)$ takes its maximum at the point $\widetilde{v}$ and for global well-definedness we would need $H(w) < 0$.

The following theorem gives a necessary and sufficient condition for global well-definedness of the `SOR` algorithm when $x < 0$. The $x = 0$ case is deferred to Theorem 3.6 because a separate analysis is needed for this simpler case.

Theorem 3.1. (global well-definedness) *Let $x < 0$. When $\omega \geq 1$, $\{v_k\}$ in the `SOR` algorithm is well-defined for any $v_0 \in \mathbb{R}_+$. When $-1 < \omega < 1$, $\{v_k\}$ is well-defined for any $v_0 \in \mathbb{R}_+$ if and only if $\omega > \widehat{\omega}$ and $\omega \geq -c_*$.*

While the global well-definedness condition guarantees that the `SOR` algorithm produces a well-defined sequence, it does not guarantee that such an ω will make the `SOR` algorithm convergent. In the following, we study the local and global convergence properties of the `SOR` algorithm.

3.3. Local convergence properties of the `SOR` algorithm

We will first analyze the local convergence properties. We say that the `SOR` algorithm **converges locally** if there exists a neighborhood V_ϵ of v_*, such that the for any $v_0 \in V_\epsilon$, the `SOR` algorithm converges. Let us introduce two functions $\Phi(v; x)$ and $\Psi(v; x)$ defined as follows

$$\Phi(v; x) = \frac{v^2 - 2|x|}{v^2 + 2|x|}, \quad \text{and} \quad \Psi(v; x) = \frac{-2|x|}{v^2 + 2|x|}. \tag{18}$$

We will write $\Phi(v)$ instead of $\Phi(v; x)$ unless we want to emphasize the dependence on x. Similarly for $\Psi(v; x)$. The two functions are related through $\Phi(v) = 1 + 2\Psi(v)$. Notice that when $x < 0$, $-1 < \Phi(v) < 1$, $-1 < \Psi(v) < 0$, and both $\Phi(v)$ and $\Psi(v)$ are strictly increasing in v. When $x = 0$, $\Phi(v) \equiv 1$ and $\Psi(v) \equiv 0$. These two functions $\Phi(v)$ and $\Psi(v)$ play extremely important roles in the analysis that follows.

The following lemma gives properties of the iteration function $G(v)$. In particular, it says that $G(v)$ is strictly increasing on any connected open subsets of $\mathrm{Dom}G \cap \{v : \omega > \Phi(v)\}$.

Lemma 3.4. *Let $x \leq 0$. Let $\mathrm{Dom}G$ denote the open set of v for which $G(v)$ is well-defined in a neighborhood of v. For $v \in \mathrm{Dom}G$, we have*

$$\frac{dG(v)}{dv} = \frac{\omega - \Phi(v)}{1 + \omega} \frac{G(v)^2\, n^+(v)}{v^2\, n^+(G(v))} \frac{v^2 + 2|x|}{G(v)^2 + 2|x|}. \tag{19}$$

Furthermore, $v_* \in \mathrm{Dom}G$, and

$$G'(v_*) = \frac{\omega - \Phi(v_*)}{1 + \omega}. \tag{20}$$

We need another technical lemma to establish the local convergence property. Define a new function $Q(v; x, v_*, \omega) : \mathrm{Dom}G \mapsto \mathbb{R}$ as follows:

$$Q(v) = Q(v; x, v_*, \omega) \equiv c(v) + c(G(v)) - 2c_*. \tag{21}$$

Notice that $Q(v_*) = 0$. Suppose $G(v)$ and $G(G(v))$ are both defined. Taking the sum of two consecutive iteration equations, we have

$$Q(v) = (1 + \omega)[N^+(v) - N^+(G(G(v))]. \tag{22}$$

Thus, the sign of $Q(v_k)$ controls whether v_{k+2} is larger than v_k or not. In particular, to help convergence, we would like to see $Q(v) > 0$ if $v > v_*$ and $Q(v) < 0$ if $v < v_*$. The following lemma gives the derivatives of $Q(v)$. It is useful in proving both Theorem 3.2 on local convergence and Theorem 3.9 on global convergence.

Lemma 3.5. *For any $v \in \mathrm{Dom}G$, the derivative of $Q(v)$ is given by*

$$Q'(v) = \frac{n^+(v)}{\frac{|x|}{G(v)^2} + \frac{1}{2}} \left[\left(\frac{|x|}{v^2} + \frac{|x|}{G(v)^2} \right) + \frac{\omega}{1 + \omega} \right]. \tag{23}$$

In particular, when $v = v_*$, we have

$$Q'(v_*) = \frac{2n^+(v_*)}{1 + \omega} \big[\omega - \Psi(v_*)\big]. \tag{24}$$

Furthermore, when $\omega = \Psi(v_*)$, we have $Q'(v_*) = Q''(v_*) = 0$ and

$$Q'''(v_*) = \frac{2n^+(v_*)|x|}{v_*^4(v_*^2 + 2|x|)}(12v_*^2 - v_*^4 + 4x^2).$$

The above two lemmas give the following theorem, which establishes conditions for the SOR algorithm to have local convergence.

Theorem 3.2. (conditions for local convergence) *Let $\omega > -1$.*

1) When $x < 0$, a necessary condition for the SOR algorithm to converge locally is $\omega \geq \Psi(v_*)$ and a sufficient condition is $\omega > \Psi(v_*)$. If $\omega = \Psi(v_*)$, a further necessary condition for local convergence is $12v_*^2 - v_*^4 + 4x^2 \geq 0$ while a further sufficient condition is $12v_*^2 - v_*^4 + 4x^2 > 0$.

2) When $x = 0$, the necessary and sufficient condition for the SOR algorithm to converge locally is $\omega > 0$.

Ordinarily, to check these conditions, we would need to know the value of v_*, which is the goal of the SOR algorithm. However, in the next section, we will introduce two acceleration techniques which employ the above theorem but avoid the knowledge of v_*. In one of the techniques, we dynamically vary ω in each iteration k, such that ω_k is eventually larger than $\Psi(v_*)$. In the other technique, we set ω to be the constant 1 in each iteration so $\omega \geq \Psi(v_*)$ is trivially satisfied.

Although we give a detailed proof of Theorem 3.2 in the Appendix, Theorem 3.2 (except for the borderline case $\omega = \Psi(v_*)$) is actually a special application of a well-known result in numerical analysis. See, for example, Isaacson and Keller (1994) for a more general statement. Indeed, in our proof we make little use of the special structure of the SOR algorithm. Notice that when $x < 0$, the condition $\omega > \Psi(v_*)$ is only sufficient but not necessary for local convergence, and the condition $\omega \geq \Psi(v_*)$ is necessary but not sufficient. This is because the borderline case $\omega = \Psi(v_*)$ is a little bit complicated. The `SOR` algorithm can either locally converge or diverge in this case. When both $\omega = \Psi(v_*)$ and $12v_*^2 - v_*^4 + 4x^2 = 0$ hold, it can be shown that $Q^{(4)}(v_*) = 0$ since in this case it is proportional to $[3v_*^6 - 6v_*^4 x + 24x^3 - 4v_*^2 x(8 + 3x)](12v_*^2 - v_*^4 + 4x^2)$. Thus, a further condition on $Q^{(5)}(v_*)$ is needed to guarantee local convergence. We do not analyze this double borderline case in more detail because we will never encounter it in our actual implementation.

The following function will play a pivotal role in the analysis of the `SOR` algorithm:

$$\phi(u,v) = \phi(u,v;x) \equiv \begin{cases} \dfrac{N^-(u) - N^-(v)}{N^+(v) - N^+(u)} & \text{if } u \neq v, \\ \Phi(u;x) & \text{if } u = v. \end{cases} \tag{25}$$

Lemma 3.6. *The function $\phi(u,v)$ is symmetric in u and v and continuous on $\mathbb{R}_+^2$. Furthermore, if $x < 0$, then $|\phi(u,v)| < 1$, and for any fixed v, $\phi(u,v)$ is continuously differentiable and strictly increasing in u, with the derivative given by*

$$\phi_1(u,v) = \begin{cases} \dfrac{n^+(u)}{[N^+(u) - N^+(v)]^2} \left\{ \dfrac{x}{u^2}\,[c(u) - c(v)] + \dfrac{1}{2}\,[c^+(u) - c^+(v)] \right\} & \text{if } u \neq v, \\ \dfrac{4\,|x|\,v}{(v^2 + 2\,|x|)^2} & \text{if } u = v. \end{cases} \tag{26}$$

Similarly, for any fixed u, $\phi(u,v)$ is continuously differentiable and strictly increasing in v, with $\phi_2(u,v) = \phi_1(v,u)$. If $x = 0$, then $\phi(u,v) \equiv 1$ on $\mathbb{R}_+^2$.

The following three lemmas are useful in analyzing both the local and global behavior of the sequence $\{v_k\}$ from the `SOR` algorithm.

Lemma 3.7. *Let $\{v_k\}$ be a well-defined `SOR` sequence and $v_k \neq v_*$ for all $k \in \mathbb{N}$.*

1) If $v_k < v_*$, then $v_{k+1} > v_k$;

2) If $v_k > v_*$, then $v_{k+1} < v_k$.

Lemma 3.8. *Let $\{v_k\}$ be a well-defined `SOR` sequence and $v_k \neq v_*$ for all $k \in \mathbb{N}$. Then for any k, we have $\omega \neq \phi(v_k, v_*)$, and*

1) If $\omega > \phi(v_k, v_*)$, then v_k and v_{k+1} are on the same side of v_*;

2) If $\omega < \phi(v_k, v_*)$, then v_k and v_{k+1} are on the opposite side of v_*.

Lemma 3.9. *Let $\{v_k\}$ be a well-defined `SOR` sequence and $v_k \neq v_*$ for all $k \in \mathbb{N}$.*

1) If $1 + 2\omega > \phi(v_k, v_{k+1})$, then v_{k+1} and v_{k+2} are on the same side of v_k;

2) If $1 + 2\omega < \phi(v_k, v_{k+1})$, then v_{k+1} and v_{k+2} are on the opposite side of v_k;

3) If $1 + 2\omega = \phi(v_k, v_{k+1})$, then $v_{k+2} = v_k$.

The next theorem states that if the SOR algorithm converges, then it is either eventually monotone or eventually oscillating around v_*.

Theorem 3.3. (local convergence pattern) *Let $x \leq 0$. Suppose that v_k from the SOR algorithm converges to v_* and $v_k \neq v_*$ for all $k \in \mathbb{N}$. Then there exists a $k_0 \in \mathbb{N}$, such that if $k > k_0$, the following is true:*

1) If $\omega \geq \Phi(v_*)$, then v_k approaches v_* monotonically;

2) If $\omega < \Phi(v_*)$, then v_k is oscillating around v_*. More specifically, for any $m \in \mathbb{N}$, we have $v_{k_0+2m} < v_{k_0+2m+2} < v_*$ and $v_{k_0+2m-1} > v_{k_0+2m+1} > v_*$.

We give two more results on the patterns of the SOR algorithm below.

Theorem 3.4. (local pattern for $|c_k - c_*|$) *Suppose that the SOR algorithm converges to v_* with $v_k \neq v_*$ for all $k \in \mathbb{N}$. Then the sequence $|c_k - c_*|$ eventually monotonically decreases to 0.*

Theorem 3.5. (local pattern for $|v_k - v_*|$) *Suppose that the SOR algorithm converges to v_* and $v_k \neq v_*$ for all $k \in \mathbb{N}$. When $\omega > \Psi(v_*)$, $|v_k - v_*|$ is eventually monotonically decreasing. When $\omega = \Psi(v_*)$, $|v_k - v_*|$ is eventually monotonically decreasing if and only if $v_* = \sqrt{2|x|}$.*

When $\omega = \Psi(v_*)$ and $v_* \neq \sqrt{2|x|}$, $|v_k - v_*|$ no longer monotonically decreases to 0. However, in this case, v_k oscillates around v_* and the two subsequences above and below v_* approach v_* monotonically.

3.4. Global convergence properties of the SOR algorithm

In the actual implementation it is hard to know whether the initial v_0 is close enough to v_* or not. Thus, in the following we study the global convergence properties. That is, in the analysis below, we do not assume that v_0 is close to v_*. Furthermore, in the majority of the analysis below, we also do not assume that the global well-definedness condition is satisfied.

The following theorem completely characterizes the well-definedness and convergence patterns for the SOR algorithm when $x = 0$. Recall from Theorem 3.2 that a necessary condition for the algorithm to converge when $x = 0$ is $\omega > 0$.

Theorem 3.6. (SOR algorithm when $x = 0$) *Assume $x = 0$ and $v_0 \neq v_*$. We have three cases:*

1) $\omega > 1$. In this case, the SOR algorithm is globally well-defined and convergent. Furthermore, if $v_0 > v_*$, then v_k strictly decreases to v_*. If $v_0 < v_*$, then v_k strictly increases to v_*.

2) $\omega = 1$. In this case, the SOR algorithm is globally well-defined and convergent. In fact, $v_k = v_*$ for any $k \geq 1$.

3) $0 < \omega < 1$. In this case, the SOR algorithm is well-defined if and only if $G(v_0)$ is defined. Furthermore, if $G(v_0)$ is defined, then the SOR algorithm is convergent and $\{v_k\}$ is immediately oscillating around v_*.

The following analysis will focus on $x < 0$. We first consider the baseline case $\omega = 0$ in the `SOR` algorithm. The following theorem shows that in this case, the `SOR` algorithm is always globally well-defined and convergent provided that v_1 is defined.

Theorem 3.7. **(`SOR` algorithm when $\omega = 0$)** *Let $x < 0$ and $\omega = 0$. Suppose $c_* + N^-(v_0) < 1$, then $\{v_k\}$ in the `SOR` algorithm is globally well-defined and converges to v_*. Furthermore, the condition $c_* + N^-(v_0) < 1$ is always satisfied when $v_* \leq \sqrt{2|x|}$, and satisfied if $v_0 \geq v_*$ or $v_0 \leq 2|x|/v_*$ when $v_* > \sqrt{2|x|}$.*

We will now consider a general relaxation parameter ω. Since a necessary condition for the `SOR` algorithm to converge locally is $\omega \geq \Psi(v_*)$, we only consider such ω's.

It turns out that when $\omega \geq \Phi(v_*)$, the `SOR` algorithm is globally convergent if the sequence is well-defined. By Theorem 3.3, the `SOR` sequence eventually monotonically approaches v_* if it converges in this case.

Theorem 3.8. **(`SOR` algorithm when $\omega \geq \Phi(v_*)$)** *Let $x < 0$ and $v_0 \neq v_*$. Assume that $\omega \geq \Phi(v_*)$. When $v_0 < v_*$, $\{v_k\}$ from the `SOR` algorithm is globally well-defined and monotonically increases to v_*. When $v_0 > v_*$, we have the following three cases:*

1) If $\omega > \phi(v_0, v_*)$, then $\{v_k\}$ is globally well-defined and monotonically decreases to v_*.

2) If $\omega = \phi(v_0, v_*)$, then $\{v_k\}$ is globally well-defined and $v_k = v_*$ for all $k \geq 1$.

3) If $\Phi(v_*) \leq \omega < \phi(v_0, v_*)$, then $\{v_k\}$ is globally well-defined if v_1 is defined. Furthermore, if v_1 is defined, then $v_1 < v_*$ and the sequence monotonically increases to v_*.

While the above theorem completely characterizes the behavior of the `SOR` algorithm when $\omega \geq \Phi(v_*)$, we do not have a complete characterization for the $\omega < \Phi(v_*)$ case. Notice that by Theorem 3.3, the `SOR` sequence is eventually oscillating around v_* if it converges when $\omega < \Phi(v_*)$.

Nevertheless, we have obtained three sufficient conditions to guarantee convergence that are possibly weaker than $\omega \geq \Phi(v_*)$. These are given in the three theorems that follow. They show that to have global convergence, we only need to slightly strengthen the local convergence condition $\omega \geq \Psi(v_*)$, which is the same as $1 + 2\omega \geq \Phi(v_*)$.

Theorem 3.9. **(first sufficient condition for global convergence)** *Let $x < 0$. Suppose that $\omega \geq \Psi(\sqrt{2}v_*)$. Let $\{v_k\}$ be the `SOR` sequence, possibly only defined for finitely many k. Let v_{k_0} be well-defined and the first point in the sequence $\{v_k\}$ such that $\omega < \phi(v_k, v_*)$. Then the whole sequence $\{v_k\}$ is well-defined if and only if $G(v_{k_0})$ is defined. Furthermore, if $G(v_{k_0})$ is defined, then the `SOR` algorithm converges to v_*.*

Whether the sequence is eventually monotone or oscillating in Theorem 3.9 depends on whether $\omega \geq \Phi(v_*)$. Also, if k_0 in the theorem does not exist, then the `SOR` algorithm is globally well-defined and converges monotonically to v_*. Another consequence of the above theorem is that since $\Psi(\sqrt{2}v_*) < 0$ when $x < 0$, the `SOR` algorithm always converges when $\omega \geq 0$ provided that $\{v_k\}$ is well-defined.

Theorem 3.10. (second sufficient condition for global convergence) *Let $x \leq 0$. Suppose that the* SOR *sequence $\{v_k\}$ is well-defined and bounded above by some $\overline{v} \geq v_*$. Then the sequence converges to v_* if $1 + 2\omega > \phi(v_*, \overline{v})$.*

For the last theorem, we first need to establish some properties of $\phi(u, G(u))$.

Lemma 3.10. *Let $x < 0$, $-1 < \omega < 1$, and $u \in \mathrm{Dom}G$. If $u < G(u)$, there exists $K > 0$, such that*

$$\phi(u, G(u))' > K(\omega - \Phi(u)).$$

As a result, when $\omega < \Phi(v_*)$, we have $\phi(u, G(u))' > 0$ if $u < \widetilde{v}$, where $\widetilde{v}$ is the unique point satisfying $\omega = \Phi(\widetilde{v})$.

Theorem 3.11. (third sufficient condition for global convergence) *Let $x \leq 0$. Suppose that ω satisfies the global well-definedness condition. Then for any $v_0 \in \mathbb{R}_+$, the* SOR *algorithm converges to v_* if*

$$1 + 2\omega > \sup_{\widetilde{v} < u < v_*} \phi(u, G(u)). \tag{27}$$

Figure 3 summarizes the results of this section by giving all the possible convergence patterns for the SOR algorithm. The top half of Figure 3 plots all the possible convergence patterns when $x = 0$. Notice that by Theorem 3.6 all the conditions above each subplot are both sufficient and necessary for that particular convergence pattern to occur. The bottom half of Figure 3 plots all the possible convergence patterns when $x < 0$. Notice that Theorem 3.8 only guarantees convergence when $\omega \geq \Phi(v_*)$. We box two conditions in two of the subplots to indicate that these conditions are only necessary and not sufficient for these two particular global convergence patterns.

4. Convergence acceleration methods

4.1. The convergence order of the SOR algorithm

We next take a look at the speed of convergence. We make use of the following common notions of convergence speed. Suppose that the sequence v_k converges to v_*. We say that this sequence converges linearly if there exists a number $\mu \in (0, 1)$ such that

$$\lim_{k \to +\infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|} = \mu. \tag{28}$$

If the above limit exists with $\mu = 0$, we say that the sequence converges superlinearly. If $\mu = 1$, we say that the sequence converges sublinearly. The number μ is usually called the convergence rate. A smaller μ means faster convergence. We say that the sequence converges with order q if for some $q \in \mathbb{N}$,

$$\lim_{k \to +\infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|^q} = \mu \text{ with } \mu > 0. \tag{29}$$

In particular, convergence with order 2 is called quadratic convergence. The following proposition shows that the `SOR` algorithm in the last section usually has a linear order of convergence.

Theorem 4.1. (convergence order of the `SOR` algorithm) *Let $x \leq 0$. Suppose that v_k from the `SOR` algorithm converges to v_* and $v_k \neq v_*$ for all $k \in \mathbb{N}$. Then,*

1) If $\omega \neq \Phi(v_*)$ and $\omega > \Psi(v_*)$, the sequence $\{v_k\}$ has a a linear order of convergence, with convergence rate $\mu = |G'(v_*)|$.

2) If $\omega = \Phi(v_*)$, then the sequence converge superlinearly.

3) If $\omega = \Psi(v_*)$, then the sequence converges sublinearly.

The above theorem immediately implies the following. Consider a fixed option with moneyness $x \leq 0$ and implied volatility v_*. Let v_k^A and v_k^B be two convergent `SOR` sequences associated with initial values v_0^A and v_0^B, and relaxation parameters ω^A and ω^B, respectively. Suppose $v_k^i \neq v_*$ for all $k \in \mathbb{N}$, $i = A, B$. If $\mu^A < \mu^B$, then Theorem 4.1 implies that regardless of the relative accuracy of v_0^A and v_0^B, there exists $k_0 \in \mathbb{N}$, such that for all $k > k_0$, $|v_k^A - v_*| < |v_k^B - v_*|$. Thus, in selecting ω, we should try to select it such that it is as close to $\Phi(v_*)$ as possible. In particular, if ω happens to be $\Phi(v_*)$, then we have superlinear convergence.

A linear convergence order is not very efficient. The rest of this section improves the convergence order of the `SOR` algorithm through two convergence acceleration techniques: dynamic relaxation and transformation of sequence.

4.2. Dynamic relaxation (`SOR-DR`)

Although the theory tells us that we should select ω to be $\Phi(v_*)$, in practice we do not know the value of v_*, because after all, the precise value of v_* is the goal of the `SOR` algorithm. This difficulty can be overcome by a dynamic relaxation technique which approximates $\Phi(v_*)$ adaptively with $\Phi(v_k)$. More specifically, we modify the `SOR` algorithm to the following, which we label as the `SOR-DR` algorithm.

```
1.  Select an initial point $v_0$ and set $\omega_0 = \Phi(v_0)$;
2.  (SOR-DR) After obtaining $v_k$, compute $v_{k+1}$ from the following equation
        $v_{k+1} = G(v_k; x, v_*, \omega_k),$        (30)
    and set $\omega_{k+1} = \Phi(v_{k+1})$;
3.  Stop when $|v_k - v_*| < \epsilon$ or $|c_k - c_*| < \epsilon$, for some $\epsilon$ small.
```

The next theorem shows that if v_1 is defined, then the `SOR-DR` algorithm always converges monotonically to v_*.

Theorem 4.2. (convergence properties of the `SOR-DR` algorithm) *When $x = 0$, the `SOR-DR` algorithm is globally well-defined and $v_k = v_*$ for all $k \in \mathbb{N}$. When $x < 0$ and $v_0 > v_*$, v_k is globally well-defined and monotonically decreases to v_*. When $x < 0$ and $v_0 < v_*$, the `SOR-DR` algorithm is globally well-defined if v_1 is defined. If v_1 is defined, then $v_1 > v_*$, and v_k monotonically decreases to v_* afterwards.*

In addition to having nice global well-definedness and convergence properties, the `SOR-DR` algorithm has at least quadratic order of convergence.

Theorem 4.3. (convergence order of the `SOR-DR` algorithm) *Assuming $v_k \neq v_*$ for all k, the `SOR-DR` algorithm has at least a quadratic order of convergence if v_1 is defined.*

Theorem 4.3 is a very interesting result, because while the Newton-Raphson algorithm uses derivative information of $c(x, v)$, the `SOR-DR` algorithm never requires the calculation of the derivatives. The derivative of $c(x, v)$ with respect to v is given by $n^+(v)$. For deeply away-from-the-money options, $n^+(v)$ could be extremely small, often resulting in overflow problems in implementing the Newton-Raphson algorithm. That the `SOR-DR` algorithm works well for away-from-the-money options is one major advantage over the Newton-Raphson method.

4.3. Transformation of sequence (`SOR-TS`)

While we can obtain quadratic order of convergence through a dynamic relaxation technique, quadratic convergence can also be obtained through a classical transformation of sequence technique. Well-known examples of sequence transformation include Aitken's delta-squared method and Richardson extrapolation. A very good up-to-date reference to this technique is Sidi (2002). We will label the following the `SOR-TS` algorithm.

1. `Select an initial point` v_0 `and a relaxation parameter` ω;
2. `(SOR-TS) After obtaining` v_k, `compute` v_{k+1} `from the following equation`

$$v_{k+1} = \alpha_k G(v_k; x, v_*, \omega) + (1 - \alpha_k) v_k, \tag{31}$$

   `where`

$$\alpha_k = \frac{1 + \omega}{1 + \Phi(v_k)}; \tag{32}$$

3. `Stop when` $|v_k - v_*| < \epsilon$ `or` $|c_k - c_*| < \epsilon$, `for some` ϵ `small.`

The iteration step in the `SOR-TS` algorithm can be more succinctly written as

$$v_{k+1} = M(v_k; x, v_*, \omega), \tag{33}$$

where the iteration function $M(v; x, v_*, \omega)$ is given by

$$M(v) = M(v; x, v_*, \omega) \equiv \frac{1 + \omega}{1 + \Phi(v; x)} G(v; x, v_*, \omega) + \left(1 - \frac{1 + \omega}{1 + \Phi(v; x)}\right) v. \tag{34}$$

We will use the common practice in numerical analysis of using the term extrapolation generally to include interpolation. That is, we treat extrapolation as a synonym for sequence transformation. The transformation of sequence technique above is a simple extrapolation using only two points v_k and $G(v_k)$. The weight α_k is chosen carefully so that $M'(v_*; x, v_*, \omega) = 0$, which guarantees at least quadratic order of convergence if the `SOR-TS` algorithm converges. Using the notions in Sidi (2002), the extrapolation is nonlinear, in that the parameter α_k is not a constant, but rather depends on the sequence $\{v_k\}$ itself. As Sidi (2002) points out, nonlinear extrapolation is usually needed to improve order of convergence. Also, the sequence transformation is iterative in that for each k, the extrapolated value v_k is used to perform the next extrapolation to get v_{k+1}.

While ω is a free parameter in the `SOR-TS` algorithm, there is some guidance on how to choose a good ω. First, ω should be chosen such that the sequence $\{v_k\}$ is well-defined. Thus, ω can not be too small. For example, if ω is close to $\Psi(v_*)$, $G(v)$ fails to be globally well-defined. Also, ω can not be too large because a too large ω might send the extrapolated point v_{k+1} below 0. The following numerical example illustrates this point. Let $x = -0.1$, $v_* = 0.1$ and $\omega = 5$. Let $v_0 = 2$. Then $G(v_0) = 1.3137$ and the extrapolated point $v_1 = -0.1617 < 0$. Another consideration in selecting ω is stability. Let $\widehat{v}_k$ be the computed numerical value for v_k. As Sidi (2002) points out, when v_k is sufficiently close to v_*, the round-off error $|\widehat{v}_k - v_k|$ might start to dominate the total error $|\widehat{v}_k - v_*|$. Furthermore, the round-off error might propagate and make the algorithm unstable if more than necessary many iterations are performed. There are a few possible sources of round-off errors. The first source is the iteration equation. This is due to the inaccuracy in computing the cumulative normal distribution and its inverse, and the effect is largely controlled by the coefficients ω and $1 + \omega$. For example, a very large ω will amplify the errors. This source of round-off error is also present in the `SOR-DR` algorithm. However, the `SOR-TS` algorithm has another source of round-off error coming from the sequence transformation. The magnitude of this source is largely controlled by the coefficients α_k and $1 - \alpha_k$.

Because of the above considerations, we recommend setting $\omega = 1$ always in the `SOR-TS` algorithm. There are several reasons for this choice. First, the choice of $\omega = 1$ is good from stability considerations. Second, if $x = 0$ and $\omega = 1$, we immediately have $G(v_1) = v_*$, and since $\alpha_1 = 1$, we have $v_1 = v_*$. That is, the sequence immediately lands on v_*. Finally, if $x < 0$ and $\omega = 1$, Theorem 3.1 guarantees the global well-definedness of $G(v)$, and Theorem 4.4 below in turn guarantees the global well-definedness of $M(v)$. The global well-definedness property is extremely useful because it brings in the robustness of the `SOR-TS` algorithm.

We first establish some useful results for the extrapolated iteration function $M(v; x, v_*, \omega)$ in Lemma 4.1 below when $\omega = 1$. These results will be used in Theorem 4.4.

Lemma 4.1. *Let $x < 0$ and $\omega = 1$. The derivative of $M(v)$ with respect to v is given by*

$$M'(v) = -\frac{4G|x|}{v^3} + \frac{2|x|}{v^2} + \frac{n^+(v)}{n^+(G)} \frac{1}{\frac{1}{2} + \frac{|x|}{G^2}} \left(\frac{|x|}{v^2} + \frac{2x^2}{v^4} \right), \tag{35}$$

where we have written G for $G(v; x, v_*, 1)$. Furthermore, $M(v) > 0$ for all $v \in \mathbb{R}_+$. Also, we have $M'(v_*) = 0$, and

$$M''(v_*) = -\frac{|x|}{2v_*^3(v_*^2 + 2\,|x|)^2} m(v_*, x)$$

where the function $m(v, x)$ is given by $m(v, x) = v^6 - 2(4 + x)v^4 - 4x^2v^2 + 8x^3$. For any fixed $x < 0$, there exists a unique $v_r = v_r(x) \in \mathbb{R}_+$ such that $m(v_r, x) = 0$. Furthermore, $m(v, x) < 0$ when $v < v_r$, and $m(v, x) > 0$ when $v > v_r$.

Since $M'(v_*) = 0$, we immediately have local convergence. Also, the sign of $M''(v_*)$ controls the convergence pattern. For global convergence, we need to define two more quantities:

$$M_{\min}(x, v_*) \equiv \inf\{M(v; x, v_*, 1) : v \geq v_*\}, \tag{36}$$

$$B(x, v_*) \equiv \sup\{|M'(v; x, v_*, 1)| : v \geq M_{\min}(x, v_*)\}. \tag{37}$$

Notice that since $M(v_*) = v_*$, we have $M_{\min}(x, v_*) \leq v_*$. We have the following theorem:

Theorem 4.4. (convergence properties of the `SOR-TS` algorithm) *Let $x \leq 0$. Then the `SOR-TS` algorithm with $\omega = 1$ is globally well-defined and $\{v_k\}$ converges to v_* locally. For any option with (x, v_*), the `SOR-TS` algorithm converges globally for any $v_0 \in \mathbb{R}_+$ if $B(x, v_*) < 1$. Furthermore, if $x = 0$, then $v_k = v_*$ for all $k \in \mathbb{N}$. If $x < 0$ and $\{v_k\}$ converges to v_* with $v_k \neq v_*$ for all $k \in \mathbb{N}$, we have*

1) The sequence $\{v_k\}$ eventually decreases to v_* if $v_* < v_r(x)$.

2) The sequence $\{v_k\}$ eventually increases to v_* if $v_* > v_r(x)$.

3) If $v_* = v_r(x)$, the convergence can be either eventually monotone or eventually oscillating.

The condition $B(x, v_*) < 1$ is only sufficient for global convergence and not necessary. Through extensive numerical analysis, we conjecture that the `SOR-TS` sequence converges globally without this condition, but so far we are not able to prove it analytically. In any case, we have numerically verified (details are available upon request) that the condition $B(x, v_*) < 1$ is satisfied inside a very large domain D^- on which we will implement our algorithms. That is, inside the domain D^-, the `SOR-TS` algorithm with $\omega = 1$ is globally convergent.

Theorem 4.5. (convergence order of the `SOR-TS` algorithm) *Let $x \leq 0$. Assuming the sequence $\{v_k\}$ from the `SOR-TS` algorithm with $\omega = 1$ converges to v_* with $v_k \neq v_*$ for all $k \in \mathbb{N}$, then the sequence has at least a quadratic order of convergence.*

Figure 4 gives all the convergence patterns for the `SOR-DR` and `SOR-TS` ($\omega = 1$) algorithms. Notice that the two algorithms are always globally well-defined, except that when $v_0 < v_*$ and $x < 0$, we need $G(v_0)$ to be well-defined to guarantee that $\{v_k\}$ is defined in the `SOR-DR` algorithm. When $x = 0$, the convergence patterns of the two algorithms are exactly the same. They both land on v_* after just one iteration. When $x < 0$, the `SOR-DR` algorithm always eventually converges to v_* monotonically from above. We classify the local convergence behavior of the `SOR-TS` algorithm according to the sign of $M''(v_*)$. The three conditions listed should be interpreted as necessary conditions. For example, a necessary condition for the `SOR-TS` sequence to eventually decreases to v_* is that $M''(v_*) \geq 0$.

While the above discussion has focused on improving the asymptotic convergence speed of the `SOR` algorithm by modifying the iteration step, in the actual implementation the initial estimate v_0 is often of crucial importance because in practice only a finite number of iterations can be performed. A good estimate v_0 usually also helps with numerical stability. Therefore, for each option with observed moneyness x and option price c_*, we use a rational approximation $v_0 = v_0(x, c_*)$ for the initial estimate. This will be discussed in detail in the next section.

5. Numerical implementation and performance

5.1. Numerical implementation with rational approximation enhancement

First, we will describe the domain of inversion. The domain of inversion consists of all options we consider, with different values of moneyness x and implied total volatility v_*. We will restrict $v_* \geq 0.0005$ since volatilities lower than this bound are extremely rare in real financial applications. Options with values c_* extremely close to 0 or 1 are also excluded. The final inversion domain D we consider is as follows:

$$D = \{\, |x| \leq 3,\ 0.0005 \leq v_* \leq 6,\ 0.0005 \leq c_* \leq 0.9995,\ |x|/v_* \leq 3 \,\}. \tag{38}$$

Notice that this domain is much larger than those considered by most authors. For example, in Li (2008), v_* is bounded above by 1, $|x|$ bounded by 0.5, and $x/|v_*|$ bounded by 2. Figure 5 plots the domain D in the two-dimensional (c_*, v_*)-space. Recall that we will only consider options with $x \leq 0$ by the "in-and-out" duality. We denote the left half of D by D^-, where $x \leq 0$. Notice we have also plotted the curve $M''(v_*) = 0$, which is the same as $v_* = v_r(x)$. By Lemma 4.1, this curve separates the left domain D^- further into two parts.

Before we look at the `SOR-DR` and `SOR-TS` algorithms in the whole domain D^-, let us look at their performance on a particular option. The option used has moneyness $x = -0.5$ and volatility $v_* = 1$. For the `SOR` and `SOR-TS` algorithms, we set $\omega = 1$ for all the iterations. Table 1 shows the effect of introducing dynamic relaxation or sequence transformation to the `SOR` algorithm. All the numbers are computed using Mathematica 6.0 with (10^{-16})-precision arithmetic, except for the rows labeled as $4'$ and $5'$, where all the numbers are computed with (10^{-50})-precision arithmetic. For all three algorithms, the initial estimate v_0 is set to be 0.6 in Panel A and 1.4 in Panel B. While all three algorithms converge to the true v_*, the `SOR-DR` and `SOR-TS` algorithms converge much faster. In each of the two latter algorithms, the number of correct digits roughly double after each iteration, indicating quadratic convergence. For the `SOR-DR` algorithm, we also give the value of ω_k for each iteration. Notice that ω_k approaches

$\Phi(v_*) = 0$ quickly, giving rise to quadratic convergence. For the `SOR-TS` algorithm, we also give the value of α_k for each iteration. As we see, α_k approaches a value of 2 quickly. Thus for each iteration, after getting $G(v_k)$, the `SOR-TS` algorithm will perform an extrapolation, which is the source of the quadratic convergence. Also notice that the round-off error kicks in and dominates the total error when v_k is extremely close to v_* if we use (10^{-16})-precision arithmetic.

We now consider the choice of $v_0 = v_0(x, c_*)$. For each option characterized by (x, c_*), we use the following third-order rational approximation for the initial estimate v_0:

$$v_0 = \frac{\sum_{i+j\le 3} m_{ij}\ x^i c_*^j}{\sum_{i+j\le 3} n_{ij}\ x^i c_*^j}, \tag{39}$$

where m_{ij} and n_{ij} are coefficients and we set $n_{00} = 1$ for normalization. Thus there are a total of 19 parameters. We search for the optimal values for $\{m_{ij}\}$ and $\{n_{ij}\}$ numerically by minimizing the uniform error in v_* from both the `SOR-DR` and `SOR-TS` algorithms. Specifically, let $v_k^{\texttt{SOR-DR}}$ and $v_k^{\texttt{SOR-TS}}$ denote the numerical estimate of v_* after k iterations for a fixed set of parameters $\{m_{ij}\}$ and $\{n_{ij}\}$ from the `SOR-DR` and `SOR-TS` algorithms, respectively. Our objective is to search for the optimal parameters through the following problem:

$$G_D = \min_{\{m_{ij}\},\{n_{ij}\}} \max_{(x,v_*)\in D^-} g(x, v_*, \{m_{ij}\}, \{n_{ij}\}), \tag{40}$$

where g is the error function for each fixed option (x, v_*) with parameters $\{m_{ij}\}$ and $\{n_{ij}\}$:

$$g(x, v_*, \{m_{ij}\}, \{n_{ij}\}) = \left|v_k^{\texttt{SOR-DR}} - v_*\right| + \left|v_k^{\texttt{SOR-TS}} - v_*\right|.$$

We have set k to be the same for all options in domain D^- for vectorization consideration. Vectorization increases the efficiency of our algorithms. To compute the max part of the above objective function, we populate the domain D^- densely with roughly one million options, with the boundary populated denser than the inner region. Each of these options is characterized by a different pair of value (x, v_*). For each fixed set of parameters $\{m_{ij}\}$ and $\{n_{ij}\}$, we compute $v_k^{\texttt{SOR-DR}}$ and $v_k^{\texttt{SOR-TS}}$, and then approximate the uniform error with the maximum error in v_* of all the options. If the domain D^- is populated sufficiently densely, this is a very accurate approximation. For the min part, we use a downhill simplex method of Nelder and Mead (1965), which is described in detail in Press et al. (1992). The choice of using the simplex method is dictated by the fact that we do not have any derivative information of the objective function with respect to the parameters $\{m_{ij}\}$ and $\{n_{ij}\}$.

A few details are worth brief mentioning. First, it is extremely difficult to minimize over uniform errors using simplex methods because uniform errors are prone to problems such as local minimums, slow convergence, etc. Thus, we actually minimize the following penalized objective function

$$G_D^\lambda = \min_{\{m_{ij}\},\{n_{ij}\}} \left(\max_{(x,v_*)\in D^-} g(x, v_*, \{m_{ij}\}, \{n_{ij}\}) + \lambda \sum_{(x,v_*)\in D^-} g(x, v_*, \{m_{ij}\}, \{n_{ij}\}) \right), \tag{41}$$

where λ controls the strength of penalty. When λ is larger, the objective function becomes smoother but deviates more from the uniform error. We dynamically adjust the value λ so that we put more weight on the uniform error as we move closer to the optimal values for $\{m_{ij}\}$ and $\{n_{ij}\}$. Second, we set $k = 5$. Numerically, we find that setting $k = 5$ uniformly for all options achieves the best combination of accuracy and efficiency. Finally, by design, the rational approximation estimate v_0 is not necessarily close to v_* for a given option (x, v_*). This is because v_k in our algorithms can "jump" and globally it is not always the case that a closer v_0 would result in a quicker convergence to v_*.

Our final choice for the $\{m_{ij}\}$ and $\{n_{ij}\}$ from the above numerical procedure is:

$$\begin{aligned}
m_{00} &= -0.00006103098165; & n_{00} &= 1; \\
m_{01} &= 5.33967643357688; & n_{01} &= 22.96302109010794; \\
m_{10} &= -0.40661990365427; & n_{10} &= -0.48466536361620; \\
m_{02} &= 3.25023425332360; & n_{02} &= -0.77268824532468; \\
m_{11} &= -36.19405221599028; & n_{11} &= -1.34102279982050; \\
m_{20} &= 0.08975394404851; & n_{20} &= 0.43027619553168; \\
m_{03} &= 83.84593224417796; & n_{03} &= -5.70531500645109; \\
m_{12} &= 41.21772632732834; & n_{12} &= 2.45782574294244; \\
m_{21} &= 3.83815885394565; & n_{21} &= -0.04763802358853; \\
m_{30} &= -0.21619763215668; & n_{30} &= -0.03326944290044.
\end{aligned} \tag{42}$$

Plugging the above parameters values into equation (39) gives us the initial starting point v_0 for the successive over-relaxation algorithms.

5.2. Numerical performance

Table 2 gives the performance of the three algorithms `SOR`, `SOR-DR`, `SOR-TS` inside the domain D^-. The accuracy is measured in terms of both $|v_k - v_*|$ and $|c_k - c_*|$. All three algorithms are implemented with (10^{-16})-precision arithmetic in MATLAB 7.1 on a Dell Dimension 4600 desktop computer (2.8 GHz, 1G RAM). We report the accuracy of the rational approximation ($k = 0$), and of each of the three algorithms when $k = 4$ and $k = 5$. The means, medians and maximums of $|v_k - v_*|$ and $|c_k - c_*|$ are calculated by uniformly and densely populating the domain D^- with roughly 1 million options. The computing time for all options of each algorithm is also reported for $k = 0, 4$ and 5. As we see, the rational approximation has a uniform error of 0.783 for $|v_k - v_*|$ and a uniform error of 0.29 for $|c_k - c_*|$ in domain D^-. Thus, the rational approximation by itself does not give accurate enough estimates for v_*. This is because we have not tried to obtain the best possible rational approximation per se in our implementation, but rather we have tried to obtained the best possible rational approximation conditioning on that we are going to perform five more successive over-relaxation iterations. The rational approximation takes only 0.65 seconds for all roughly one million options. When $k = 4$, both the accelerated algorithms `SOR-DR` and `SOR-TS` achieve a uniform error in v_* in the order of 10^{-8} while that of the `SOR` algorithm is 0.078. The medians of the errors in v_* and c_* are quite small for all three algorithms. The larger errors tend to occur near the boundary of the domain D^-. All three algorithms take around 12 seconds. For $k = 5$, both accelerated algorithms `SOR-DR` and `SOR-TS` achieve a uniform error in v_* in the order of 10^{-13} while that of the `SOR` algorithm is 0.058. We do not recommend using more than 5 iterations because the round-off error can start to dominate the total error in v_*. The computing times are still only about 12 seconds for all roughly one million options. Overall, we see that the performance of the `SOR-DR` and `SOR-TS` algorithms are quite good in terms of both accuracy and speed. Also, numerically we find that our algorithms work for a much larger domain with only a slight decrease in accuracy.

Most existing methods fail to work in our large domain D^-. This is the case, for example, for the Corrado-Miller method and the rational approximation of Li (2008). Thus we cannot perform a comparison between these methods and ours. One exception is the Dekker-Brent algorithm. Table 3 compares the performance of the `SOR-TS` algorithm, the Newton-Raphson algorithm, and the Dekker-Brent algorithm for a particular option (x, v_*), where we let $x = -1$ and $v_* = 2$. We do not consider the `SOR-DR` algorithm in this comparison, because its performance is very similar to that of the `SOR-TS` algorithm, except for the fact that the `SOR-DR` algorithm is not always globally well-defined. We start all three algorithms from three different initial values, namely, 0.1, 4, and 20 and report the first 5 iterations (Panel A) together with the errors (Panel B). As we see, a naive implementation of the Newton-Raphson algorithm fails in all three cases. The reason is that in each Newton-Raphson iteration, one needs to divide the price error by the vega in each step. For away-from-the-money options, vega can be extremely small, leading to numerical instability. Although not reported, for this particular option, if we start from the the guess $v_0 = \sqrt{2|x|}$ as was suggested in Manaster and Koehler (1982), then the Newton-Raphson algorithm converges. However, examples can be easily given where even this choice of $v_0 = \sqrt{2|x|}$ leads to failure in the Newton-Raphson algorithm. This happens for large values of $|x|$ and v_* in our domain D^-. On the other hand, both the `SOR-TS` and Dekker-Brent algorithms converge in all three cases. The quadratic order of convergence of the `SOR-TS` algorithm is evident after only one or two iterations. The Dekker-Brent algorithm uses a combination of bisection and interpolation and some of the bisection steps are evident in the table. Overall, the `SOR-TS` algorithm achieves the best combination of robustness, efficiency and accuracy. One particular attractive feature of this algorithm is that it is globally convergent for any positive v_0.

6. Further Applications

The idea of using successive over-relaxation to compute the implied volatility is useful in other financial applications. Below we give three examples to demonstrate this point. In our first example, we consider the implied correlation for an exchange option in the Margrabe framework.

In the second example, we consider the implied volatility for call options in the Bachelier model. The last example considers the critical stock price level in a compound call on call option.

6.1. The Margrabe implied correlation

The most commonly used formula for the price of an exchange option is the Margrabe formula, which was independently discovered by Fischer (1978) and Margrabe (1978). Margrabe (1978) considers the price of an option to exchange one asset for another, while Fischer (1978) considers the price of a call option when the exercise price is uncertain.

Under the Margrabe formula setup, the dynamics of the two stock prices $S_1(t)$ and $S_2(t)$ under the risk-neutral measure $\mathbb{Q}$ are given by

$$\mathrm{d}S_i(t) = (r - \delta_i)S_i(t)\mathrm{d}t + \sigma_i S_i(t)\mathrm{d}W_i(t), \quad (i = 1, 2)$$

where the two Brownian motions $W_1(t)$ and $W_2(t)$ are correlated with constant coefficient ρ. The Margrabe formula gives the time-0 price C of a European option to exchange stock 2 for stock 1 at time T as follows:

$$C = S_1(0)e^{-\delta_1 T}N(d_1) - S_2(0)e^{-\delta_2 T}N(d_2),$$

where

$$d_1 = \frac{\log[(S_1(0)e^{-\delta_1 T})/(S_2(0)e^{-\delta_2 T})] + \sigma^2 T/2}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T},$$

and $\sigma \equiv \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$. Li (2007) contains three different methods to prove the Margrabe formula.

In practice, people often use the Margrabe formula to compute the implied correlation ρ from the observed option price C, with σ_1 and σ_2 estimated using some other methods. Our methods can be applied directly to compute the implied correlation. This is because if we define

$$c = C/(S_1 e^{-\delta_1 T}), \quad x = \log[(S_1(0)e^{-\delta_1 T})/(S_2(0)e^{-\delta_2 T})], \quad v = \sigma\sqrt{T}, \tag{43}$$

then the Margrabe formula reduces to the dimensionless Black-Scholes formula in (1). The parameter v can then be computed using either the `SOR-DR` or the `SOR-TS` algorithm. Once v is computed, the implied correlation can be computed by

$$\rho = \frac{\sigma_1^2 T + \sigma_2^2 T - v^2}{2\sigma_1\sigma_2 T}. \tag{44}$$

6.2. The Bachelier implied volatility

We now consider the Bachelier formula (Bachelier, 1900) for a European call option. This formula was discovered by L. Bachelier, who was among the first to analyze Brownian motion mathematically. Schachermayer and Teichmann (2007) contain an interesting comparison of Bachelier's formula with that of Black and Scholes. The original Bachelier formula considers interest rate $r = 0$. There are a few approaches to extend it to nonzero interest rates. We adopt the following one which models the risk-neutral dynamics of the stock price process as:

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma e^{(r-\delta)t}\mathrm{d}W_t.$$

Notice that we have introduced a nonzero dividend rate δ too. In this case, S_T has the explicit solution $S_T = e^{(r-\delta)T}(S_0 + \sigma W_T)$, and the Bachelier formula for the time-0 price of a call option is given by

$$C = \sigma\sqrt{T}e^{-\delta T}n(d) + (S_0e^{-\delta T} - Ke^{-rT})N(d), \tag{45}$$

where

$$d = \frac{S_0e^{-\delta T} - Ke^{-rT}}{\sigma\sqrt{T}e^{-\delta T}}.$$

In practice, people observe the actual price C_* and compute the volatility parameter σ_* which satisfies the Bachelier formula. This parameter (after sometimes normalizing it by S_0) is called the Bachelier implied volatility. It is also often called the normal volatility by practitioners because future stock price is assumed to be normally distributed in the Bachelier model.

Recently, Choi, Kim and Kwak (2007) develop a closed-form numerical approximation for the Bachelier implied volatility, which is found to be very accurate. In this paper, we will apply our successive over-relaxation method to compute the Bachelier implied volatility. We first make the following parameter reduction. Define the normalized call price c, moneyness x and volatility v as follows:

$$c = C/(S_0e^{-\delta T}), \quad x = 1 - \frac{Ke^{-rT}}{S_0e^{-\delta T}}, \quad v = \sigma\sqrt{T}/S_0. \tag{46}$$

Then Bachelier's formula in equation (45) reduces to the following dimensionless Bachelier's formula

$$c = c^B(x, v) = v\, n(x/v) + xN(x/v). \tag{47}$$

Notice that like the Black-Scholes formula, the Bachelier option price is also a strictly increasing function of v because the Bachelier vega is given by $\partial c^B(x, v)/\partial v = n(x/v) > 0$.

Like the dimensionless Black-Scholes formula, we only need to consider call options because put-call parity is also satisfied in the Bachelier framework. Also, we only need to consider out-of-the-money call options because of the following "in-out" duality in the Bachelier formula:

$$c^B(x, v) - c^B(-x, v) = x.$$

That is, to compute the Bachelier implied volatility for an option with positive moneyness x and normalized price c_*, we could just compute the volatility for its dual option with moneyness $-x$ and normalized option price $c_* - x$.

Let v_* be the true Bachelier implied volatility. That is, $c_* = c^B(x, v_*)$. Define the iteration function $G^B(v; x, v_*, \omega)$ as follows:

$$G^B(v; x, v_*, \omega) = \frac{c_* - xN(x/v) + \omega v n(x/v)}{(1+\omega)n(x/v)}. \tag{48}$$

We will sometimes just write $G^B(v)$. One possible successive over-relaxation method is the following, which we label as the `SOR-B` algorithm:

```
1. Select an initial point $v_0$;
2. (SOR-B) After obtaining $v_k$, select $\omega_k$ and compute $v_{k+1}$ by
```

$$v_{k+1} = G^B(v_k; x, v_*, \omega_k), \tag{49}$$

```
3. Stop when $|v_k - v_*| < \epsilon$ or $|c_k - c_*| < \epsilon$, for some $\epsilon$ small.
```

It turns out that with the choice $\omega_k = 0$, the `SOR-B` algorithm is globally well-defined and converges to v_* with a quadratic order of convergence.

Theorem 6.1. (convergence properties of the `SOR-B` algorithm) *Let $\omega_k \equiv 0$ in the `SOR-B` algorithm. Then the sequence $\{v_k\}$ is globally well-defined. If $x = 0$, then $v_k = v_*$ for all $k \geq 1$. If $x < 0$ and $v_0 \neq v_*$, then $v_1 > v_*$ and v_k monotonically decreases to v_* afterwards with a quadratic order of convergence.*

The efficiency of the `SOR-B` algorithm can be improved by introducing a rational approximation like the one we did for the Black-Scholes implied volatility case. We omit the details here. Another way to extend the Bachelier model to nonzero interest rate is to model the stock price process as $\mathrm{d}S_t = (r - \delta)S_t\mathrm{d}t + \sigma\mathrm{d}W_t$. In this case, a different formula from equation (45) will be obtained. See, Musiela and Rutkowski (2005). However, our method of using successive over-relaxation to compute the Bachelier implied volatility is still applicable after some minor modifications.

6.3. The critical stock price in a compound option

Compound options are options on options. Their pricing and various applications are considered by Geske (1977), Geske (1979), Hodges and Selby (1987), and Chandrasekhar and Gukhal (2004), among others. Below we consider a call on call option in the standard Black-Scholes framework. We will design a successive over-relaxation method to compute the critical stock price. For other types of compound options, such as call on put, put on put and put on call, our method still applies with minor modifications.

Let the current date be 0, and T_1 and T_2 be two future dates with $T_2 > T_1 > 0$. We will let $\tau = T_2 - T_1$. Let the current stock price be S_0, the constant interest rate be r, the dividend rate be δ, and the volatility of the stock be σ. At time T_1, the holder of the compound option is entitled to receive either cash K_1, or a call option with strike price K_2 and maturity date T_2. The time-0 value C^c of the compound option is given by

$$C^c = S_0 e^{-qT_2} N_2(a_1, b_1; \sqrt{T_1/T_2}) - K_2 e^{-rT_2} N_2(a_2, b_2; \sqrt{T_1/T_2}) - e^{-rT_1} K_1 N(a_2),$$

where $N_2(\,\cdot\,, \cdot\,; \rho)$ is the cumulative bivariate normal distribution function with correlation ρ, and

$$a_1 = \frac{\log(S_0/S_*) + (r - q + \sigma^2/2)T_1}{\sigma\sqrt{T_1}}, \qquad a_2 = a_1 - \sigma\sqrt{T_1},$$
$$b_1 = \frac{\log(S_0/K_2) + (r - q + \sigma^2/2)T_2}{\sigma\sqrt{T_2}}, \qquad b_2 = b_1 - \sigma\sqrt{T_2}.$$

The formula for C^c is in closed form except that the critical stock price level S_* at time T_1 to exercise the compound option is given implicitly by

$$K_1 = S_* e^{-q\tau} N\left(\frac{1}{\sigma\sqrt{\tau}} \log \frac{S_* e^{-q\tau}}{K_2 e^{-r\tau}} + \frac{\sigma\sqrt{\tau}}{2}\right) - K_2 e^{-r\tau} N\left(\frac{1}{\sigma\sqrt{\tau}} \log \frac{S_* e^{-q\tau}}{K_2 e^{-r\tau}} - \frac{\sigma\sqrt{\tau}}{2}\right).$$

A Newton-Raphson algorithm can be used to compute S_*, but it is subject to the same numerical instability we encountered before. To implement a successive over-relaxation method, we first perform a dimension reduction. Defining modified strike κ, moneyness x, and total volatility v by

$$\kappa = \frac{K_1}{K_2 e^{-r\tau}}, \quad x = \log \frac{S_* e^{-q\tau}}{K_2 e^{-r\tau}}, \quad v = \sigma\sqrt{\tau}, \tag{50}$$

the critical stock price equation becomes

$$\kappa = \kappa(x, v) = e^x N(x/v + v/2) - N(x/v - v/2). \tag{51}$$

The problem is then to compute the moneyness x_* that satisfies the above equation with observed modified strike κ_* and assumed value of the total volatility v.

Notice that $\kappa_* = \kappa(x_*, v)$. Define the iteration function $G^{CS}(x; v, x_*, \omega)$ by

$$G^{CS}(x; v, x_*, \omega) = \log\left(\frac{\kappa(x_*, v) + N(x/v - v/2) + \omega e^x N(x/v + v/2)}{(1 + \omega) N(x/v + v/2)}\right). \tag{52}$$

One possible successive over-relaxation method is the following `SOR-CS` algorithm:

```
1.  Select an initial point $x_0$;
2.  (SOR-CS) After obtaining $x_k$, select $\omega_k$ and compute $x_{k+1}$ by
```

$$x_{k+1} = G^{CS}(x_k; v, x_*, \omega_k); \tag{53}$$

```
3.  Stop when $|x_k - x_*| < \epsilon$ or $|\kappa_k - \kappa_*| < \epsilon$, for some $\epsilon$ small.
```

Notice that unlike the Newton-Raphson algorithm, in the `SOR-CS` algorithm, no evaluation of the standard normal density function $n(\cdot)$ is required. The following theorem shows that the `SOR-CS` algorithm is globally well-defined and converges to x_* with a quadratic order of convergence.

Theorem 6.2. (convergence properties of the `SOR-CS` algorithm) *Let $x_* \in \mathbb{R}$ and $v > 0$. Let $\omega_k \equiv 0$ in the SOR-CS algorithm. Then the sequence $\{x_k\}$ is globally well-defined. Furthermore, if $x_0 \neq x_*$, then $x_1 > x_*$ and x_k monotonically decreases to x_* afterwards with a quadratic order of convergence.*

Although we do not give detailed report here, numerically we find that the above algorithm converges extremely fast and is very robust. One critical advantage of the above method versus Newton-Raphson is that it is not very sensitive to the initial guess x_0 for all reasonable values of x_* and v. Also, a rational approximation can be employed on the initial value x_0 to enhance convergence. We omit the details here.

7. Conclusion

The Black-Scholes formula is one of the most frequently used formula in finance. In most of the applications, it is used backwards to compute the implied volatility. Some traditional methods, such as the Newton-Raphson algorithm, use derivative information and are thus subject to numerical instability. Numerically more stable methods, such as the Dekker-Brent algorithm, are usually slow and often not able to handle the real-time computation of the implied volatility in bulk volume.

In this paper, we design a successive over-relaxation (`SOR`) algorithm which does not use the derivative information. We analyze the well-definedness, local and global convergence properties of the `SOR` algorithm in detail. With mild conditions, the algorithm is globally well-defined and converges to the true implied volatility.

The `SOR` algorithm generally has a linear order of convergence. By introducing dynamic relaxation or sequence transformation techniques, both new algorithms `SOR-DR` and `SOR-TS` achieve quadratic order of convergence, the same order as the Newton-Raphson algorithm. The efficiency of the algorithms are further enhanced by introducing a rational approximation on the initial estimates. It is shown that uniformly in a very large inversion domain, our accelerated algorithms converge to the true implied volatility with very few iterations. Thus, the accelerated algorithms serve as good alternatives to the traditional methods because of their good performance in terms of speed, accuracy and robustness.

Finally, we extend our successive over-relaxation method to the computation of implied correlation in the Margrabe formula, the normal implied volatility in the Bachelier formula, and the critical stock price level in a compound option. These three examples demonstrate that the idea of successive over-relaxation is applicable in a much wider range of financial problems.

Appendix

Proof of Lemma 3.1:

For $x < 0$, we have

$$\begin{aligned}
\frac{\partial N^+(x,v)}{\partial v} &= n^+(v)\left(\frac{|x|}{v^2}+\frac{1}{2}\right),\\
\frac{\partial N^-(x,v)}{\partial v} &= n^+(v)\left(\frac{|x|}{v^2}-\frac{1}{2}\right),\\
\frac{\partial c(x,v)}{\partial v} &= n^+(v), \qquad \frac{\partial^2 c(x,v)}{\partial v^2} = \frac{n^+(v)}{v}\left(\frac{|x|^2}{v^2}-\frac{v^2}{4}\right),\\
\frac{\partial c^+(x,v)}{\partial v} &= n^+(v)\frac{2\,|x|}{v^2},
\end{aligned}$$

where we have used the identity $e^{-x}n(x/v-v/2)=n(x/v+v/2)$ in the second equation. These prove statements 1 through 4. For statement 5, assume $u<v$ without loss of generality. Thus

$$c(v)-c(u)=\int_u^v \frac{\partial c^+(x,\xi)}{\partial \xi}\frac{\xi^2}{2\,|x|}d\xi < \frac{v^2}{2\,|x|}\int_u^v \frac{\partial c^+(x,\xi)}{\partial \xi}d\xi = \frac{v^2}{2\,|x|}\left[c^+(v)-c^+(u)\right].$$

Similarly for the left-hand inequality in equation (9).

For $x=0$,all the statements can be quickly verified. □

Proof of Lemma 3.2:

We have $F'(v)=n^+(v)\left(\frac{|x|}{v^2}+\frac{1}{2}\right)[\omega-\Phi(v)]$, and $\omega=\Phi(v)$ if and only if $v=\widetilde{v}(\omega)$. Thus, if $\omega\geq 1$, then $F'(v)>0$. If $\omega\in(-1,1)$, then $\widetilde{v}$ satisfies $\omega=\Phi(\widetilde{v})$. Since $\Phi(v)$ is strictly increasing in v, we have $F'(v)>0$ for $v<\widetilde{v}$ and $F'(v)<0$ for $v>\widetilde{v}$. □

Proof of Lemma 3.3:

Since $\omega=\Phi(\widetilde{v})$, we have $H'(\omega)=N^+(\widetilde{v})-1$. Hence, $H(\omega)$ is strictly decreasing in ω for $\omega\in(-1,1)$. By Lemma 3.1, we have $\lim_{\omega\to-1}H(\omega)=c_*>0$, and $\lim_{\omega\to+1}H(\omega)=c_*-1<0$. Therefore, there exists a unique $\widehat{\omega}$ such that $H(\omega)\geq 0$ for $\omega\leq\widehat{\omega}$ and $H(\omega)<0$ for $\omega>\widehat{\omega}$. □

Proof of Theorem 3.1:

Note that v_{k+1} is well-defined if and only if $0<F(v_k)<1+\omega$.

By Lemma 3.2, when $\omega\geq 1$, $F(v)$ is positive, strictly increasing in v, and bounded above by $F(+\infty)=c_*+\omega$, which is less than $1+\omega$. Hence, $\{v_k\}$ is well-defined for any $v_0\in\mathbb{R}_+$.

When $\omega\in(-1,1)$, by Lemma 3.2 and 3.3, we know that if $\omega>\widehat{\omega}$, then for any $v\in\mathbb{R}_+$,

$$c_*+N^-(v)+\omega N^+(v)-(1+\omega)\leq H(\omega)<0.$$

By Lemma 3.2, for all $v \in \mathbb{R}_+$, we also have

$$F(v) = c_* + N^-(v) + \omega N^+(v) > \min\left\{F(0^+), F(+\infty)\right\} = \min\{c_*, c_* + \omega\}.$$

Thus, if $\omega \geq -c_*$ additionally, we have $0 < F(v) < 1 + \omega$. This guarantees that $\{v_k\}$ is well-defined for any $v_0 \in \mathbb{R}_+$.

Conversely, if $\omega < -c_*$, then $F(+\infty) < 0$, so for sufficiently large v_0, we have $F(v_0) < 0$ and $v_1 = G(v_0)$ is not defined. If $\omega \leq \widehat{\omega}$, then if $v_0 = \widetilde{v}(\omega)$, we have $F(v) - (1+\omega) = H(\omega) \geq 0$. So $v_1 = G(v_0)$ is not defined. $\square$

Proof of Lemma 3.4:
Take any $v \in \mathrm{Dom}G$. Equation (19) can be obtained by differentiating

$$c_* + N^-(v) + \omega N^+(v) = (1+\omega) N^+(G(v))$$

with respect to v and using Lemma 3.1. Since $0 < F(v_*) < 1 + \omega$, for v sufficiently close to v_*, we also have $0 < F(v) < 1 + \omega$, thus $v_* \in \mathrm{Dom}G$. For equation (20), notice that $G(v_*) = v_*$. All the other statements can be quickly verified. $\square$

Proof of Lemma 3.5:
Since $Q'(v) = n^+(v) + n^+(G(v))G'(v)$ for $v \in \mathrm{Dom}G$, substituting the expression for $G'(v)$ in Lemma 3.4 gives us the formula for $Q'(v)$. The higher-order derivatives of $Q(v)$ when $\omega = \Psi(v_*)$ can be obtained by recursively differentiating $Q'(v)$ and using Lemma 3.4. $\square$

Proof of Theorem 3.2:
First consider the case $x < 0$. Since $v_* \in \mathrm{Dom}G$ by Lemma 3.4, there exists a neighborhood $V_\epsilon = (v_* - \epsilon, v_* + \epsilon)$ of v_* such that $G(v)$ is smooth on V_ϵ and $M \equiv \sup\{|G''(v)|/2 : v \in V_\epsilon\} < \infty$. Suppose $\omega < \Psi(v_*)$ and v_k converges to v_*. Then, since $G'(v_*) = [\omega - \Phi(v_*)]/(1+\omega)$ is strictly increasing in ω, we have $G'(v_*) < (\Psi(v_*) - \Phi(v_*))/(1 + \Psi(v_*)) = -1$. Pick a $k_0 \in \mathbb{N}$ large such that $|v_k - v_*| < \min\{\epsilon, \delta\}$, where $\delta = (|G'(v_*)| - 1)/(2M)$. Then, since $v_{k+1} - v_* = G(v_k) - G(v_*)$, a Taylor expansion gives $|(v_{k+1} - v_k) - G'(v_*)(v_k - v_*)| \leq M(v_k - v_*)^2$. That is,

$$\begin{aligned} |v_{k+1} - v_*| &\geq \left|G'(v_*)\right| |v_k - v_*| - M(v_k - v_*)^2 \\ &\geq |v_k - v_*| + \left(\left|G'(v_*)\right| - 1 - M\delta\right) |v_k - v_*| > |v_k - v_*|. \end{aligned}$$

This contradicts that v_k converges to v_*. Thus, a necessary for local convergence is $\omega \geq \Psi(v_*)$.

Now, suppose $\omega = \Psi(v_*)$, $12v_*^2 - v_*^4 + 4x^2 < 0$, and v_k converges to v_*. Then, by Lemma 3.5,

$$Q'(v_*) = Q''(v_*) = 0 \quad \text{and} \quad Q'''(v_*) < 0.$$

Thus, there exists a neighborhood $V_\epsilon = (v_* - \epsilon, v_* + \epsilon)$ of v_* such that $Q(v) \geq 0$ on $(v_* - \epsilon, v_*]$ and $Q(v) \leq 0$ on $[v_*, v_* + \epsilon)$. Since $Q(v_k) = (1+\omega)\left[N^+(v_k) - N^+(v_{k+2})\right]$, we have that

$v_{k+2} < v_k$ if $v_k < v_*$, and $v_{k+2} > v_k$ if $v_k > v_*$. This contradicts that v_k converges to v_*. So, a further necessary condition $12v_*^2 - v_*^4 + 4x^2 \geq 0$ is needed for local convergence. The proof that $12v_*^2 - v_*^4 + 4x^2 > 0$ is sufficient is similar.

We now show that $\omega > \Psi(v_*)$ is sufficient for local convergence. Notice that $|G'(v_*)| < 1$. Pick any number $\lambda \in (|G'(v_*)|, 1)$. Let $\delta = \frac{\lambda - |G'(v_*)|}{2M}$. Then, if $|v_k - v_*| < \min\{\epsilon, \delta\}$, we have

$$\begin{aligned} |v_{k+1} - v_*| &\leq \left|G'(v_*)\right| |v_k - v_*| + M |v_k - v_*|^2 \\ &\leq \left(\left|G'(v_*)\right| + M\delta\right) |v_k - v_*| < \lambda |v_k - v_*| . \end{aligned}$$

So, v_k converges to v_*.

Next, we consider the case $x = 0$. A very similar proof to the above shows that $\omega \geq 0$ is necessary and $\omega > 0$ is sufficient for local convergence. However, when $\omega = 0$ and $v_0 \neq v_*$, we always have $v_1 \neq v_*$, and the sequence oscillates on the set $\{v_0, v_1\}$. Thus, if $v_0 \neq v_*$, the sequence $\{v_k\}$ will never converge when $\omega = 0$. □

Proof of Lemma 3.6:

First consider the case $x < 0$. Before proving that $\phi(u,v)$ is continuous on $\mathbb{R}_+^2$, we will first prove that for any fixed v, $\phi(u,v)$ is continuous and strictly increasing in u and $\phi_1(u,v)$ is also continuous in u. Take any $v \in \mathbb{R}_+$. When $u \neq v$, differentiating $\phi(u,v)$ with respect to u gives

$$\phi_1(u,v) = \frac{n^+(u)}{\left[N^+(u) - N^+(v)\right]^2} \left\{ \frac{x}{u^2} \left[c(u) - c(v)\right] + \frac{1}{2} \left[c^+(u) - c^+(v)\right] \right\}. \tag{54}$$

By Lemma 3.1, $\phi_1(u,v) > 0$ when $u \neq v$. When $u = v$, by L′Hospital's rule, we have

$$\phi_1(v,v) \equiv \lim_{w \to v} \frac{\phi(w,v) - \Phi(v)}{w - v} = \lim_{w \to v} \phi_1(w,v).$$

Thus $\phi_1(u,v)$ is continuous in u for all $u \in \mathbb{R}_+$. Applying L′Hospital's rule two more times to equation (54), we have

$$\begin{aligned} \phi_1(v,v) &= n^+(v) \lim_{w \to v} \frac{\frac{x}{w^2}\left[c(w) - c(v)\right] + \frac{1}{2}\left[c^+(w) - c^+(v)\right]}{\left[N^+(w) - N^+(v)\right]^2} \\ &= \frac{\frac{|x|}{v^3}}{\frac{|x|}{v^2} + \frac{1}{2}} \lim_{w \to v} \frac{c(w) - c(v)}{N^+(w) - N^+(v)} = \frac{4\,|x|\, v}{(v^2 + 2\,|x|)^2} > 0. \end{aligned}$$

The partial derivative of $\phi(u,v)$ with respect to v can be easily obtained by noticing that $\phi(u,v) = \phi(v,u)$. Since $\phi(u,v)$ is strictly increasing in both u and v, we have

$$-1 < \Phi(\min(u,v)) \leq \phi(u,v) \leq \Phi(\max(u,v)) < 1.$$

Finally, we show that $\phi(u,v)$ is continuous on $\mathbb{R}_+^2$. If $u \neq v$, the continuity is obvious. So we assume $u = v$. Note first that $\Phi(v) = \phi(v,v)$ is continuous, strictly increasing in v, so for any

$\epsilon > 0$, there exists a $\delta > 0$ such that $|a - v| < \delta$ implies $|\phi(a,a) - \phi(v,v)| < \epsilon/4$. Consider an open disc that centers at (v, v) and has a diameter of δ. Then, by triangular inequality, for any (a, b) in the disc, we have

$$\begin{aligned} |\phi(a,b) - \phi(v,v)| &\leq |\phi(a,b) - \phi(a,v)| + |\phi(a,v) - \phi(v,v)| \\ &< 2\,|\phi(v+\delta/2, v+\delta/2) - \phi(v-\delta/2, v-\delta/2)| < \epsilon. \end{aligned}$$

For the case $x = 0$, notice that $c^+(u) \equiv 1$ on $\mathbb{R}_+$ by Lemma 3.1, so $\phi(u,v) \equiv 1$. □

Proof of Lemma 3.7:
Rearranging the iteration equation $c_* + N^-(v_k) + \omega N^+(v_k) = (1+\omega)N^+(v_{k+1})$, we have

$$c(v_*) - v(v_k) = (1+\omega)\left[N^+(v_{k+1}) - N^+(v_k)\right].$$

Since both $c(v)$ and $N^+(v)$ are strictly increasing in v by Lemma 3.1, $v_* - v_k$ always have the same sign as $v_{k+1} - v_k$ when $v_k \neq v_*$. □

Proof of Lemma 3.8:
Rearranging the iteration equation $c_* + N^-(v_k) + \omega N^+(v_k) = (1+\omega)N^+(v_{k+1})$, we have

$$\left(N^+(v_k) - N^+(v_*)\right)\Big[\omega - \phi(v_k, v_*)\Big] = (1+\omega)\left[N^+(v_{k+1}) - N^+(v_*)\right].$$

Since $v_{k+1} \neq v_*$, we have $\omega \neq \phi(v_k, v_*)$. Furthermore, if $\omega > \phi(v_k, v_*)$, then v_k and v_{k+1} are on the same side of v_*. If $\omega < \phi(v_k, v_*)$, then v_k and v_{k+1} are on the opposite side of v_*. □

Proof of Lemma 3.9:
Subtracting two consecutive iteration equations, we get

$$\left(N^+(v_{k+1}) - N^+(v_k)\right)\Big[(1+2\omega) - \phi(v_k, v_{k+1})\Big] = (1+\omega)\left[N^+(v_{k+2}) - N^+(v_k)\right].$$

Hence, if $1 + 2\omega > \phi(v_k, v_{k+1})$, then v_{k+1} and v_{k+2} are on the same side of v_k. If $1 + 2\omega < \phi(v_k, v_{k+1})$, then v_{k+1} and v_{k+2} are on the opposite side of v_k. If $1 + 2\omega = \phi(v_k, v_{k+1})$, then $v_{k+2} = v_k$. □

Proof of Theorem 3.3:
We only consider the case $x < 0$. The case $x = 0$ can be similarly shown by noticing that the necessary and sufficient condition for local convergence when $x = 0$ is $\omega > 0$.

First we consider $\omega > \Phi(v_*)$. By Lemma 3.6, there exists an $\epsilon > 0$ such that $\omega > \phi(v, v_*)$ for all $v \in (v_* - \epsilon, v_* + \epsilon)$. Since v_k converges to v_*, there exists a $k_0 \in \mathbb{N}$ such that $v_k \in (v_* - \epsilon, v_* + \epsilon)$ for all $k \geq k_0$. Suppose $v_{k_0} < v_*$. Then, by Lemmas 3.8 and 3.9, $v_k < v_{k+1} < v_*$ for all $k \geq k_0$. Thus, v_k $(k \geq k_0)$ strictly increases to v_*. Similarly, if $v_{k_0} > v_*$, then v_k $(k \geq k_0)$ strictly decreases to v_*

Next, suppose $\omega = \Phi(v_*)$. Since $\phi(v, v_*)$ is strictly increasing in v and $\phi(v_*, v_*) = \Phi(v_*)$ by Lemma 3.6, there exists an $\epsilon > 0$ such that $\omega > \phi(v, v_*)$ for all $v \in (v_* - \epsilon, v_*)$ and $\omega < \phi(v, v_*)$ for all $v \in (v_*, v_* + \epsilon)$. If $v_{k_0} < v_*$, then we get $v_k < v_{k+1} < v_*$ for all $k \geq k_0$, so the sequence is eventually monotonically increasing. If $v_{k_0} > v_*$, then we have $v_{k_0+1} < v_*$ by Lemma 3.8 and the sequence is eventually monotonically increasing too.

Now suppose $\omega < \Phi(v_*)$. By Lemma 3.6, there exists an $\epsilon > 0$ such that $\omega < \phi(v, v_*)$ for all $v \in V_\epsilon \equiv (v_* - \epsilon, v_* + \epsilon)$. Since v_k converges to v_*, by Lemma 3.7 and 3.8, there exists $k_1 \in \mathbb{N}$, such that $v_{k_1+m} \in V_\epsilon$, with $v_{k_1+2m} < v_*$ and $v_{k_1+2m+1} > v_*$ for all $m \in \mathbb{N}$. We are left to show that the subsequences $\{v_{k_1+2m}\}$ and $\{v_{k_1+2m+1}\}$ indexed by m are both eventually monotone. For this purpose, consider the function $Q(v)$ defined in equation (21). Recall $Q(v_k) = (1+\omega)[N^+(v_k) - N^+(v_{k+2})]$. Notice that for any $\delta > 0$, $Q(v)$ cannot be constant on any interval $[v_*, v_* + \delta)$ or $(v_* - \delta, v_*]$, because then we will have $v_k = v_{k+2m}$ for all $m \in \mathbb{N}$ for some k sufficiently large and this contradicts that v_k converges to v_*. Since the function $Q(v)$ is continuous and $Q(v_*) = 0$, there exists a $\delta > 0$, such that $Q(v)$ has determinate signs on $(v_* - \delta, v_*)$ and $(v_*, v_* + \delta)$. There are four possible cases but the only one that does not contradict convergence is $Q(v) > 0$ if $v \in (v_*, v_* + \delta)$ and $Q(v) < 0$ if $v \in (v_* - \delta, v_*)$. All other three cases will result in a subsequence of v_k which does not converge to v_*. This shows that the subsequences $\{v_{k_1+2m}\}$ and $\{v_{k_1+2m+1}\}$ are both eventually monotone. The proof is now complete by selecting $k_0 > k_1$ sufficiently large and $v_{k_0} < v_*$. □

Proof of Theorem 3.4:

By Theorem 3.3, if $\{v_k\}$ converges, the sequence is either eventually monotone or eventually oscillating around v_*. It is obvious that $|c_k - c_*|$ eventually decreases to zero monotonically if $\{v_k\}$ is eventually monotone. In the case of oscillating convergence, consider the function $Q(v)$. The proof of Theorem 3.3 establishes that to guarantee convergence, there exists a $\delta > 0$, such that $Q(v) > 0$ on $(v_*, v_*+\delta)$ and $Q(v) < 0$ on $(v_*-\delta, v_*)$. Thus, for sufficiently large k, if $v_k > v_*$, then $|c_{k+1} - c_*| - |c_k - c_*| = -Q(v_k) < 0$. If $v_k < v_*$, then $|c_{k+1} - c_*| - |c_k - c_*| = Q(v_k) < 0$. In any case, $|c_k - c_*|$ is eventually strictly decreasing. □

Proof of Theorem 3.5:

By Theorem 3.3, if $\{v_k\}$ converges, the sequence is either eventually monotone or eventually oscillating around v_*. It is obvious that $|v_k - v_*|$ eventually decreases to zero monotonically if $\{v_k\}$ is eventually monotone. Thus we only consider the case $\omega < \Phi(v_*)$, where $\{v_k\}$ is oscillating around v_* if $k \geq k_0$. Without loss of generality, we assume $v_{k_0} < v_*$. Thus, we have $v_{k_0} < v_{k_0+2} < \cdots < v_* < \cdots < v_{k_0+3} < v_{k_0+1}$.

Define a function $J(v)$ as

$$J(v) = c_* + N^-(v) + \omega N^+(v) - (1+\omega)N^+(2v_* - v). \tag{55}$$

Then, $J(v_*) = 0$, and

$$J'(v_*) = n^+(v_*)\frac{v_*^2 + 2\,|x|}{v_*^2}\left(\omega - \Psi(v_*)\right).$$

First consider the case $\omega > \Psi(v_*)$. In this case, $J'(v_*) > 0$. Hence, there exists an $\epsilon > 0$ such that $J(v) < 0$ for $v \in (v_* - \epsilon, v_*)$ and $J(v) > 0$ for $v \in (v_*, v_* + \epsilon)$. Suppose $v_k \in (v_* - \epsilon, v_* + \epsilon)$ for $k \geq k_0$. Then, for points to the right of v_*, we have

$$\begin{aligned}
&(1+\omega)\left[N^+(v_{k_0+2m+1}) - N^+(2v_* - v_{k_0+2m})\right] \\
&= c_* + N^-(v_{k_0+2m}) + \omega N^+(v_{k_0+2m}) - (1+\omega)N^+(2v_* - v_{k_0+2m}) \\
&= J(v_{k_0+2m}) < 0,
\end{aligned} \tag{56}$$

and hence $v_{k_0+2m+1} < 2v_* - v_{k_0+2m}$. That is, $|v_{k_0+2m+1} - v_*| < |v_{k_0+2m} - v_*|$. Similarly, for points to the left of v_*,

$$(1+\omega)\left[N^+(v_{k_0+2m+2}) - N^+(2v_* - v_{k_0+2m+1})\right] = J(v_{k_0+2m+1}) > 0, \tag{57}$$

and hence $|v_{k_0+2m+2} - v_*| < |v_{k_0+2m+1} - v_*|$. Therefore, $|v_k - v_*|$ strictly decreases to 0 after $k \geq k_0$.

Now consider the case $\omega = \Psi(v_*)$. In this case, $J(v_*) = J'(v_*) = 0$, and

$$J''(v_*) = \frac{n^+(v_*)}{4v_*^3}(v_*^4 - 4x^2).$$

If $v_* \neq \sqrt{2|x|}$, then either $J''(v_*) < 0$ or $J''(v_*) > 0$. Thus, on sufficiently small neighborhood of v_*, we have either $J(v) \geq 0$, or $J(v) \leq 0$. Equations (56) and (57) tell us that in either case, $|v_k - v_*|$ fails to be monotonically decreasing. If $v_* = \sqrt{2|x|}$, we need the third-order derivative of $J(v)$:

$$J'''(v_*) = \frac{3}{\sqrt{2\pi}v_*^2} > 0.$$

Hence, there exists an $\epsilon > 0$ such that $J(v) < 0$ for $v \in (v_* - \epsilon, v_*)$ and $J(v) > 0$ for $v \in (v_*, v_* + \epsilon)$, and equations (56) and (57) again hold. Thus when $\omega = \Psi(v_*)$, $|v_k - v_*|$ is eventually monotonically decreasing if and only if $v_* = \sqrt{2|x|}$. □

Proof of Theorem 3.6:

When $x = 0$, we have $\Phi(v) = 1$ and $\phi(u, v) = 1$ for all $(u, v) \in \mathbb{R}_+^2$.

1) If $\omega > 1$, then the `SOR` algorithm is globally well-defined by Theorem 3.1. Furthermore, since $\omega > \phi(u, v)$ for all $(u, v) \in \mathbb{R}_+^2$, by Lemmas 3.7 and 3.8, if $v_0 > v_*$, then v_k strictly decreases, and if $v_0 < v_*$, v_k strictly increases. By Lemma 3.1, it is easy to show that the limit points have to be v_*.

2) If $\omega = 1$, then the SOR algorithm is globally well-defined by Theorem 3.1. In fact, since $\omega = \phi(v_0, v_*)$, by Lemma 3.8, we have $v_k = v_*$ for all $k \geq 1$.

3) If $0 < \omega < 1$, then we have $\omega < \phi(u, v)$ and $1 + 2\omega > \phi(u, v)$ for all $(u, v) \in \mathbb{R}^2_+$. Hence, as long as $G(v_0)$ is defined, the SOR algorithm is well-defined. By Lemma 3.9, the two subsequences above and below v_* are both monotone. Thus, they converge to two limit points, say v_L and v_H, with $v_L \leq v_*$ and $v_H \geq v_*$. We need to show that $v_L = v_H = v_*$. Suppose $v_L < v_H$. Taking a limit on the iteration equation for odd and even k, we have

$$c_* + N^-(v_L) + \omega N^+(v_L) = (1 + \omega)N^+(v_H),$$
$$c_* + N^-(v_H) + \omega N^+(v_H) = (1 + \omega)N^+(v_L).$$

Subtracting the above two equations gives us $1 + 2\omega = \phi(v_L, v_H) = 1$. This contradicts that $\omega > 0$. Thus, the sequence $\{v_k\}$ is convergent and eventually oscillating around v_*. □

Proof of Theorem 3.7:
We first focus on the well-definedness. Suppose $v_* \leq \sqrt{2\,|x|}$. If $v \geq v_*$, then

$$c_* + N^-(v) = N^+(v_*) - N^-(v_*) + N^-(v) \leq c^+(v) < 1,$$

and if $v < v_*$, then

$$c_* + N^-(v) = N^+(v_*) - N^-(v_*) + N^-(v) < N^+(v_*) < 1.$$

Hence, $0 < c_* + N^-(v) < 1$ for all $v \in \mathbb{R}_+$.

Suppose $v_* > \sqrt{2\,|x|}$ and $c_* + N^-(v_0) < 1$. Then,

$$0 < c_* + N^-(v_1) = N^+(v_1) - N^-(v_0) + N^-(v_1) < c^+(v_1) < 1.$$

Hence, v_2 is well-defined. By induction, the whole sequence $\{v_k\}$ in the SOR algorithm is well-defined. Furthermore, if either $v_0 \geq v_*$ or $v_0 \leq 2|x|/v_*$, then $N^-(v) \leq N^-(v_*)$ and we have $0 < c_* + N^-(v) \leq N^+(v_*) < 1$.

Let us turn to the proof of convergence, assuming that the sequence $\{v_k\}$ is well-defined. Note that $\omega = 0$ in this case, and thus $1 + 2\omega > \phi(u, v)$ for all $(u, v) \in \mathbb{R}^2_+$ by Lemma 3.6. We consider three cases.

1) $v_* < \sqrt{2\,|x|}$. Then, we have $0 = \omega > \Phi(v_*)$. Suppose first $v_0 > v_*$. If $\omega > \phi(v_0, v_*)$, then, by Lemma 3.8, we have a strictly decreasing, convergent sequence. If $\omega < \phi(v_0, v_*)$, then we have $v_1 < v_*$. From here, since $\omega > \Phi(v_*) > \phi(v_1, v_*)$, we have a strictly increasing, convergent sequence. Suppose now $v_0 < v_*$. Since $\omega > \Phi(v_*) > \phi(v_0, v_*)$, we have a strictly increasing, convergent sequence.

2) $v_* = \sqrt{2\,|x|}$. Then, we have $0 = \omega = \Phi(v_*)$. Since $\phi(v, v_*)$ is strictly increasing in v and $\phi(v_*, v_*) = \Phi(v_*)$ by Lemma 3.6, we have $\omega > \phi(v, v_*)$ for all $v < v_*$ and $\omega < \phi(v, v_*)$ for all

$v > v_*$. This guarantees that $\{v_k\}$ is eventually monotonically increasing. It is easy to show that the limit point is v_*.

3) $v_* > \sqrt{2\,|x|}$. Then, since $0 = \omega < \Phi(v_*)$, there exists an $\epsilon > 0$ such that $\omega < \phi(v, v_*)$ for all $v \in (v_* - \epsilon, v_* + \epsilon)$. First suppose $v_0 < v_*$. Let v_{k_0} be the first point in $\{v_k\}$ such that $\omega < \phi(v_k, v_*)$. Such a k_0 exists since $\omega < \Phi(v_*)$ and if $\omega > \phi(v, v_*)$, then we have $v < G(v) < v_*$. Then, since we always have $1 + 2\omega > \phi(u, v)$, by Lemma 3.9, we have

$$v_{k_0} < v_{k_0+2} < \cdots < v_* < \cdots < v_{k_0+3} < v_{k_0+1}.$$

Let $v_L = \lim_m v_{k_0+2m}$ and $v_H = \lim_m v_{k_0+2m+1}$. Suppose $v_L < v_H$. Then, from the iteration equation, we have $c_* + N^-(v_L) = N^+(v_H)$ and $c_* + N^-(v_H) = N^+(v_L)$. Subtracting these two equations leads to $c^+(v_L) = c^+(v_H)$. This is a contradiction since c^+ is strictly increasing. Thus, $v_L = v_H = v_*$. That is, v_k converges to v_*. Now suppose $v_0 > v_*$. Then, since $0 = \omega < \phi(v_0, v_*)$, we immediately have $v_1 < v_*$. Thus, the proof reduces to the above with the role of v_0 replaced by v_1. □

Proof of Theorem 3.8:
When $v_0 < v_*$, we have $\omega > \phi(v_0, v_*)$ and hence we have $v_0 < v_1 < v_*$. We still have $\omega > \phi(v_1, v_*)$, and thus $v_1 < v_2 < v_*$. Hence, by induction, v_k strictly increases to v_*.

When $v_0 > v_*$,

1) If $\omega > \phi(v_0, v_*)$, then we have $v_* < v_1 < v_0$. At the second step, we still have $\omega > \phi(v_1, v_*)$, and thus $v_* < v_2 < v_1$. Hence, v_k strictly decreases to v_*.

2) If $\omega = \phi(v_0, v_*)$, then, by Lemma 3.8, $v_1 = v_*$ and thus $v_k = v_*$ for all $k \geq 1$.

3) If $\Phi(v_*) \leq \omega < \phi(v_0, v_*)$, then we have $v_1 < v_*$ if v_1 is defined. Since $\omega > \phi(v_1, v_*)$, we have $v_1 < v_2 < v_*$ and we still have $\omega > \phi(v_2, v_*)$, and thus $v_* < v_2 < v_3$. Hence, v_k strictly increases to v_*. □

Proof of Theorem 3.9:
By Theorem 3.8, we know that when $\omega \geq \Phi(v_*)$, as long as the first jump over v_* (in this case always $G(v_0)$) is well-defined, the `SOR` algorithm is well-defined and converges to v_*. Hence, we will consider only the case $\omega < \Phi(v_*)$ in the following proof.

We first establish a very useful result. When $\omega \geq \Psi(\sqrt{2}v_*)$, each iteration of the successive over-relaxation will decrease the distance of c_k to c_*, provided that v_{k+1} is well-defined. To show this, take any $v \in \mathrm{Dom}G$. Suppose that v and $G(v)$ are on the opposite side of v_*. If $\omega \geq \Psi(\sqrt{2}v_*)$, then

$$\left(\frac{|x|}{v^2} + \frac{|x|}{G(v)^2}\right) + \frac{\omega}{1+\omega} > \frac{|x|}{v_*^2} + \frac{\omega}{1+\omega} > 0.$$

Thus, $Q'(v) > 0$ by Lemma 3.5. Since $Q(v_*) = 0$, $Q(v) > 0$ if $v > v_*$ and $Q(v) < 0$ if $v < v_*$, provided that v and $G(v)$ are on the opposite side of v_*. Regardless of whether $v > v_*$ or $v < v_*$, by the definition of $Q(v)$, this guarantees

$$|c(G(v)) - c_*| < |c(v) - c_*| \tag{58}$$

whenever $(v - v_*)(G(v) - v_*) < 0$. Suppose now that v and $G(v)$ are on the same side of v_*. Then by Lemma 3.7, $|G(v) - v_*| < |v - v_*|$, so equation (58) is still true.

We now consider the well-defined of the sequence $\{v_k\}$. For any v with $\omega > \phi(v, v_*)$, it is easy to show that $G(v)$ is always well-defined and lies in the region between v and v_*. If $\omega = \phi(v, v_*)$, then $G(v)$ is also well-defined and equals v_*. Now let v_{k_0} be the first point in $\{v_k\}$ such that $\omega < \phi(v_k, v_*)$ and suppose $G(v_{k_0})$ is defined. We need to show that v_k is also well-defined for any $k > k_0 + 1$. Notice that by Lemma 3.8, $(v_{k_0} - v_*)(v_{k_0+1} - v_*) < 0$. Suppose first $v_{k_0} < v_*$. Then $v_{k_0+1} > v_*$. From $\omega < \phi(v_{k_0+1}, v_*)$, we have

$$F(v_{k_0+1}) = c_* + N^-(v_{k_0+1}) + \omega N^+(v_{k_0+1}) < (1+\omega)N^+(v_*) < 1 + \omega.$$

From equation (58), we have $2c_* - c_{k_0} - c_{k_0+1} > 0$. Thus,

$$\begin{aligned} F(v_{k_0+1}) &= c_* + N^-(v_{k+1}) + \omega N^+(v_{k_0+1}) = c_* - c_{k_0+1} + (c_* + N^-(v_{k_0}) + \omega N^+(v_{k_0})) \\ &= 2c_* - c_{k_0} - c_{k_0+1} + (1+\omega)N^+(v_{k_0}) > (1+\omega)N^+(v_{k_0}) > 0. \end{aligned}$$

Thus, $v_{k_0+2} = G(v_{k_0+1})$ is well-defined and $v_{k_0} < v_{k_0+2} < v_*$. Similarly, if $v_{k_0} > v_*$, then $v_{k_0+2} = G(v_{k_0+1})$ is well-defined and $v_{k_0} > v_{k_0+2} > v_*$. Exactly the same proof now shows that v_k is well-defined for any $k > k_0 + 2$ and lies in between v_{k-1} and v_*. This shows that the whole sequence $\{v_k\}$ is well-defined if and only if $G(v_{k_0})$ is defined.

Now assume that $\{v_k\}$ is well-defined. We know the two subsequences consisting of points above v_* and below v_* respectively are both monotone. Thus they converge to two limit points, say v_H and v_L. Taking the limit on the iteration equation for odd and even k then gives $v_L = G(v_H)$ and $v_H = G(v_L)$. Suppose $v_H > v_L$. Notice $v_L = G(G(v_L))$. This is a contradiction, however, because since $|G(G(v_L)) - v_*|$ has to be smaller than $|v_L - v_*|$ by equation (58), we must have $G(G(v_L)) > v_L$. Thus $v_H = v_L$ and the sequence $\{v_k\}$ converges. $\square$

Proof of Theorem 3.10:
By Theorem 3.6, we only consider $x < 0$. Furthermore, by Theorem 3.8, we need only to consider the case $\omega < \Phi(v_*)$.

If $v_0 > v_*$, then $v_1 < v_*$ since $\omega < \Phi(v_*) < \phi(v_0, v_*)$. Thus without loss of generality, we assume $v_0 < v_*$. We consider three cases.

1) If $\omega < \phi(v_0, v_*)$, then we have $v_1 > v_*$. Clearly, we still have $\omega < \phi(v_1, v_*)$ and thus $v_2 < v_*$. By the assumption $1 + 2\omega > \phi(v_*, \overline{v})$, we have $1 + 2\omega > \phi(v_0, v_1)$ since $\phi(u, v)$ is increasing in both u and v by Lemma 3.6. This guarantees $v_0 < v_2 < v_*$ by Lemma 3.9. Hence, we still have $\omega < \phi(v_2, v_*)$ and thus $v_3 > v_*$. Again, we have $1 + 2\omega > \phi(v_1, v_2)$ and hence $v_* < v_3 < v_1$. Repeating this process gives us $v_0 < v_2 < \cdots < v_* < \cdots < v_3 < v_1$. Let

$v_L = \lim_k v_{2k}$ and $v_H = \lim_k v_{2k+1}$. Suppose $v_L < v_H$. Then, taking the limit on the iteration equation for both odd and even k, we have

$$\begin{aligned} c_* + N^-(v_L) + \omega N^+(v_L) &= (1+\omega)N^+(v_H), \\ c_* + N^-(v_H) + \omega N^+(v_H) &= (1+\omega)N^+(v_L). \end{aligned}$$

Subtracting these two equations leads to $1 + 2\omega = \phi(v_L, v_H)$. This contradicts the assumption $1 + 2\omega > \phi(v_*, \overline{v})$. Hence, we have $v_L = v_H = v_*$.

2) If $\omega = \phi(v_0, v_*)$, then we have $v_k = v_*$ for all $k \geq 1$.

3) If $\omega > \phi(v_0, v_*)$, then we have $v_0 < v_1 < v_*$. Let k_0 be the first k such that $\omega < \phi(v_k, v_*)$. Then, starting from v_{k_0}, the proof reduces to that of case 1. $\square$

Proof of Lemma 3.10:
We will write v for $G(u)$. Suppose $u < v$. By Lemma 3.4 and 3.6, we have

$$\begin{aligned} \phi(u, G(u))' &= \phi_1(u, G(u)) + \phi_2(u, G(u))G'(u) \\ &= \frac{n^+(u)}{[N^+(u) - N^+(v)]^2} \times \Bigg\{ (c(v) - c(u)) \Bigg[-\frac{|x|}{v^2} \frac{\left(\frac{|x|}{u^2} - \frac{1}{2}\right) + \omega \left(\frac{|x|}{u^2} + \frac{1}{2}\right)}{(1+\omega)\left(\frac{|x|}{v^2} + \frac{1}{2}\right)} + \frac{|x|}{u^2} \Bigg] \\ &\quad + \frac{1}{2}(c^+(v) - c^+(u)) \Bigg[-1 + \frac{\left(\frac{|x|}{u^2} - \frac{1}{2}\right) + \omega \left(\frac{|x|}{u^2} + \frac{1}{2}\right)}{(1+\omega)\left(\frac{|x|}{v^2} + \frac{1}{2}\right)} \Bigg] \Bigg\} \\ &= M \left\{ (c(v) - c(u)) \left[-\frac{|x|}{v^2} A + \frac{|x|}{u^2} \right] + \frac{1}{2} \left(c^+(v) - c^+(u)\right) [-1 + A] \right\}, \end{aligned}$$

where $M = n^+(u) / \left[N^+(u) - N^+(v)\right]^2 > 0$, and

$$A = \frac{\left(\frac{|x|}{u^2} + \frac{1}{2}\right)}{(1+\omega)\left(\frac{|x|^2}{v^2} + \frac{1}{2}\right)} \left[\omega - \Phi(u)\right].$$

Since $u < v$ and $1 + \omega > 0$, it can be easily shown that $-\frac{|x|}{v^2}A + \frac{|x|}{u^2} > 0$. Using the inequality between c and c^+ in Lemma 3.1, we have

$$\begin{aligned} \phi(u, G(u))' &\geq M \left\{ \frac{u^2}{2|x|} \left(c^+(v) - c^+(u)\right) \left[-\frac{|x|}{v^2} A + \frac{|x|}{u^2} \right] + \frac{1}{2} \left(c^+(v) - c^+(u)\right) [-1 + A] \right\} \\ &= \frac{M}{2} \left(c^+(v) - c^+(u)\right) \left[1 - \frac{u^2}{v^2} \right] A \geq K \left(\omega - \Phi(u)\right). \end{aligned}$$

Notice that $\widetilde{v}$ is finite when $\omega < \Phi(v_*)$. Furthermore, $\widetilde{v} < v_*$ since $\Phi(v)$ is strictly increasing. Since $u < \widetilde{v} < v_*$, by Lemma 3.7, we have $G(u) > u$. Also, we have $\omega = \Phi(\widetilde{v}) > \Phi(u)$. Thus, $\phi(u, G(u))' > 0$. $\square$

Proof of Theorem 3.11:
When $x = 0$, the condition in the theorem becomes $\omega > 0$ and guarantees convergence by Theorem 3.6. Thus we assume $x < 0$. Furthermore, we assume $\omega < \Phi(v_*)$ since if $\omega \geq \Phi(v_*)$, then Theorem 3.8 guarantees convergence.

By Lemma 3.10, $\phi(u, G(u))$ is increasing in u when $u < \widetilde{v}$. Thus, $\sup_{u<\widetilde{v}} \phi(u, G(u)) = \phi(\widetilde{v}, G(\widetilde{v}))$. The condition in the theorem then implies that $1 + 2\omega > \sup_{u<v_*} \phi(u, G(u))$. By Lemma 3.2, we have $G(\widetilde{v}) = \max_{v \in \mathrm{Dom} G} G(v)$.

If $v_0 > v_*$, then $v_1 < v_*$ since $\omega < \Phi(v_*) < \phi(v_0, v_*)$. Thus without loss of generality, we assume $v_0 < v_*$. Since the sequence obviously converges if $v_k = v_*$ for some k, we will assume $v_k \neq v_*$ for all k. This in particular implies that $v_k \neq v_{k+1}$ for all k, and $\omega \neq \phi(v_k, v_*)$ for any k. We consider two cases.

1) $\omega < \phi(v_0, v_*)$. We have $v_1 > v_*$. Since $\omega < \phi(v_1, v_*)$, $v_2 < v_*$. By equation (7), we have $1 + 2\omega > \phi(v_0, v_1)$, so $v_0 < v_2 < v_*$ by Lemma 3.9. Now since $\omega < \phi(v_2, v_*)$, we have $v_3 > v_*$. Since $\omega < \phi(v_3, v_*)$, we get $v_4 < v_*$. Again, because $1 + 2\omega > \phi(v_2, v_3)$, $v_2 < v_4 < v_*$. Repeating this process gives us $v_0 < v_2 < v_4 < \cdots < v_*$, and $v_* < v_{2k+1} < G(\widetilde{v})$ for all $k \geq 0$. We now show v_{2k+1} is decreasing. Suppose $v_3 > v_1$. Since $G(v)$ is strictly decreasing for $v > \widetilde{v}$ by Lemma 3.4, we get $v_4 = G(v_3) < G(v_1) = v_2$. This is a contradiction. Hence, we also have $v_* < \cdots < v_5 < v_3 < v_1$. Let $v_L = \lim_k v_{2k}$ and $v_H = \lim_k v_{2k+1}$. Suppose $v_L < v_H$. Similar argument as in the proof of Theorem 3.6 gives that when $v_L \neq v_H$, we have $1 + 2\omega = \phi(v_L, v_H)$. This contradicts equation (7). Hence, we have $v_L = v_H = v_*$.

2) $\omega > \phi(v_0, v_*)$. We have $v_0 < v_1 < v_*$. Let k_0 be the first k such that $\omega < \phi(v_k, v_*)$. Notice k_0 is finite by Lemma 3.7 and the fact that $\omega < \Phi(v_*)$. For otherwise, v_k would strictly increase to a limit point other than v_*, which is impossible since v_* is the only fixed point of $v = G(v)$. The proof now reduces exactly to the previous case, with v_{k_0} playing the role of v_0. □

Proof of Theorem 4.1:
From the proof of Theorem 3.2, we have

$$v_{k+1} - v_* = G(v_k) - G(v_*) = G'(v_*)(v_k - v_*) + \mathcal{O}(v_k - v_*)^2.$$

Since v_k converges to v_*, we have

$$\lim_{k \to +\infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|} = \left|G'(v_*)\right|.$$

Notice that $\omega \geq \Psi(v_*)$ implies that $|G'(v_*)| \leq 1$. By Lemma 3.4, we have $G'(v_*) = 0$ if and only if $\omega = \Phi(v_*)$. Also, $|G'(v_*)| = 1$ if and only if $\omega = \Psi(v_*)$. Thus, $\{v_k\}$ has sublinear convergence when $\omega = \Psi(v_*)$, superlinear convergence when $\omega = \Phi(v_*)$, and linear convergence in all other cases. □

Proof of Theorem 4.2:

When $x = 0$, by Lemma 3.6, we have $\omega_0 = \Phi(v_0) = \Phi(v_*) = \phi(v_0, v_*) = 1$ for any $v_0 \in \mathbb{R}_+$. Hence, by using a similar proof as in Lemma 3.8, we have that $v_k = v_*$ for all $k \in \mathbb{N}$.

Now suppose $x < 0$. When $v_0 > v_*$, by Lemma 3.7 and 3.8, we have that v_1 is well-defined and $v_* < v_1 < v_0$ since $\omega_0 = \Phi(v_0) > \phi(v_0, v_*)$. Again, since $\omega_1 = \Phi(v_1) > \phi(v_1, v_*)$, we have that v_2 is well-defined and $v_* < v_2 < v_1$. Hence, v_k strictly decreases to v_*.

When $v_0 < v_*$, by Lemma 3.8 we have $v_1 > v_*$ if v_1 is well-defined, since $\omega_0 = \Phi(v_0) < \phi(v_0, v_*)$. The proof now reduces to the previous case, with v_1 playing the role of v_0. □

Proof of Theorem 4.3:

By Theorem 4.2, v_k converges to v_* from above if v_1 is well-defined. Thus $\omega_k = \Phi(v_k)$ converges to $\Phi(v_*)$. Let us write $G(v_k; \omega_k)$ for $G(v_k; x, v_*, \omega_k)$. Since $v_* \in \mathrm{Dom}G$ by Lemma 3.4, there exists a neighborhood $V_\epsilon = (v_* - \epsilon, v_* + \epsilon)$ of v_* such that $G(v; \omega)$ is smooth on V_ϵ and

$$M_1(\omega) \equiv \sup\left\{\left|G''(v; \omega)\right|/2 : v \in V_\epsilon\right\} < \infty,$$

for each fixed ω. Also, since $G(v; \omega)$ is smooth in ω, we can assume

$$M_2 = \sup\left\{\left|G''(v; \omega)\right|/2 : v \in V_\epsilon,\ \omega \in \Omega_\epsilon\right\} < \infty,$$

where $\Omega_\epsilon = (\Phi(v_*) - \epsilon, \Phi(v_*) + \epsilon)$.

Now, Taylor expansion gives us

$$v_{k+1} - v_* = G(v_k; \omega_k) - G(v_*; \omega_k) = G'(v_k; \omega_k)(v_k - v_*) + \frac{1}{2}G''(\xi_k, \omega_k)(v_k - v_*)^2,$$

where $\xi_k \in (v_k, v_*)$. Hence, using L'Hospital's rule, we have

$$\begin{aligned}\lim_{k\to\infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|^2} &\le \lim_{k\to\infty} \frac{|G'(v_k; \omega_k)|}{|v_k - v_*|} + M_2 = \lim_{k\to\infty} \frac{|\omega_k - \Phi(v_*)|/(1 + \omega_k)}{|v_k - v_*|} + M_2 \\ &= \frac{|\Phi'(v_*)|}{1 + \Phi(v_*)} + M_2 = \frac{4|x|}{v_*(v_*^2 + 2|x|)} + M_2 < \infty.\end{aligned}$$

This shows that the convergence is of at least quadratic order. □

Proof of Lemma 4.1:

Notice that when $\omega = 1$, we have

$$M(v) = \frac{2}{1 + \Phi(v)}G(v) + \left(1 - \frac{2}{1 + \Phi(v)}\right)v = \frac{L(v)}{1 + \Phi(v)}, \tag{59}$$

where $L(v) = 2G(v) + (\Phi(v) - 1)\,v$. The derivative of $M(v)$ with respect to v follows directly from Lemma 3.4.

We now show that $M(v) > 0$ for all $v \in \mathbb{R}_+$. Fix $v > 0$. Note that $G'(u) > 0$ for all $u \in \mathbb{R}_+$, and $L'(u) > 0$ for all $u \ge \sqrt{2|x|}$ since

$$L'(u) = 2G'(u) + \frac{4|x|(u^2 - 2|x|)}{(u^2 + 2|x|)^2}. \tag{60}$$

Notice also $M(v_*) = v_* > 0$ and $L(v_*) > 0$. Thus, we assume $v \neq v_*$.

Suppose first $v < v_*$. Then, we have $v < G(v) < v_*$ and thus $M(v) > G(v) > 0$.

Next, we consider $v > v_*$. We prove $M(v) > 0$ by dividing the proof into two cases: $\sqrt{2|x|} \le v_*$ and $v_* < \sqrt{2|x|}$.

Case 1: $\sqrt{2\,|x|} \le v_*$. Since $L'(u) > 0$ for all $u \ge \sqrt{2\,|x|}$, we have

$$M(v) = \frac{L(v)}{1+\Phi(v)} \ge \frac{L(v_*)}{1+\Phi(v)} > 0.$$

Case 2: $v_* < \sqrt{2\,|x|}$. There are two subcases to consider. First, suppose $v_* < v \le \sqrt{2\,|x|}$. By equation (35), we have

$$M'(u) = K\left[-2G^3u + G^2u^2 + \frac{n^+(u)}{n^+(G)}G^2u^2 + 2\frac{n^+(u)}{n^+(G)}G^2\,|x| - 4Gu\,|x| + 2u^2\,|x|\right],$$

where $G = G(u)$ and $K > 0$. Since $u > G(u)$ and $n^+(u) > n^+(G)$ for all $u \in (v_*, \sqrt{2\,|x|}]$, we have

$$\begin{aligned} M'(u) &\ge K\Big[-2G^3u + 2G^2u^2 + 2G^2\,|x| - 4Gu\,|x| + 2u^2\,|x|\Big] \\ &= K\Big[2G^2u(u-G) + 2\,|x|\,(u-G)^2\Big] > 0 \end{aligned}$$

for all $u \in (v_*, \sqrt{2\,|x|}]$. Therefore, we have $M(v) > M(v_*) > 0$ for all $v_* < v \le \sqrt{2\,|x|}$.

Second, suppose $v_* < \sqrt{2\,|x|} < v$. Since $L'(u) > 0$ for all $u > \sqrt{2\,|x|}$ by equation (60), we have $L(v) > L(\sqrt{2\,|x|})$. Hence,

$$M(v) \ge \frac{L(\sqrt{2\,|x|})}{1+\Phi(v)} = \frac{M(\sqrt{2\,|x|})}{1+\Phi(v)} > \frac{M(v_*)}{1+\Phi(v)} > 0,$$

where we have used the fact that $M(\sqrt{2\,|x|}) > M(v_*)$ in the proof of the previous case.

That $M'(v_*) = 0$ follows directly from equation (35) and the fact that $G(v_*) = v_*$. The second order derivative $M''(v_*)$ follows from direct computation and Lemma 3.4.

We now examine the function $m(v, x)$. Notice first that

$$\lim_{v\to 0^+} m(v,x) = 8x^3 < 0, \quad \lim_{v\to+\infty} m(v,x) = +\infty.$$

Furthermore, we have

$$\frac{\partial m(v,x)}{\partial v} = 2v\big(-4x^2 - 4(4+x)v^2 + 3v^4\big) = 6v(v^2 - z_1)(v^2 - z_2),$$

where

$$z_1 = \frac{2}{3}\left[4 + x + 2\sqrt{(x+1)^2+3}\right] > 0, \quad z_2 = \frac{2}{3}\left[4 + x - 2\sqrt{(x+1)^2+3}\right] < 0.$$

Thus, for fixed $x < 0$, $m(v, x)$ as a function of v is strictly decreasing on $(0, \sqrt{z_1})$, and strictly increasing on $(\sqrt{z_1}, +\infty)$. By taking into account the limiting behavior of $m(x, v)$, we see that $m(v, x) = 0$ has a unique root $v_r = v_r(x) > \sqrt{z_1}$. Furthermore, $m(v, x) < 0$ when $v < v_r$, and $m(v, x) > 0$ when $v > v_r$. □

Proof of Theorem 4.4:

The case with $x = 0$ can be easily verified, so we assume $x < 0$ below.

By Lemma 4.1, when $\omega = 1$, for any $v \in \mathbb{R}_+$, $M(v)$ is well-defined and $M(v) > 0$. Thus the `SOR-TS` algorithm is globally well-defined. Since $M'(v_*) = 0$, $M(v)$ is locally contracting near v_* since $M(v)$ is smooth function of v. Hence, if v_0 is sufficiently close to v_* then the sequence $\{v_k\}$ converges to v_*.

We now show that the `SOR-TS` algorithm converges globally when $B(x, v_*) < 1$. Assume $v_k \neq v_*$ for all $k \in \mathbb{N}$. We consider the following two cases.

Case 1: $v_0 > v_*$. In this case, the sequence $\{v_k\}$ is bounded below by $M_{\min}(x, v_*)$. But then the assumption $B(x, v_*)$ implies that $M(v)$ is globally contracting on $[M_{\min}(x, v_*), \infty)$, since for any $v \in [M_{\min}(x, v_*), \infty)$, there exists ξ between v and v_*, such that

$$|M(v) - v_*| = |M'(\xi)| \cdot |v - v_*| < |v - v_*|.$$

Thus, $\{v_k\}$ converges to v_*.

Case 2: $v_0 < v_*$. In this case, we have $M(v_0) > G(v_0) > v_0$. If the sequence $\{v_k\}$ is bounded above by v_*, then $v_k < v_{k+1} < v_*$ for all $k \in \mathbb{N}$. Thus v_k monotonically increases. It is easy to show that it converges to v_*. The other case is when there exists $k_0 \in \mathbb{N}$ such that $v_{k_0} > v_*$, which reduces to Case 1 above.

The convergence pattern follows from the following Taylor expansion

$$v_{k+1} - v_* = \frac{1}{2} M''(v_*)(v_k - v_*)^2 + \mathcal{O}(v_k - v_*)^3. \tag{61}$$

Notice that when $v_* < v_r(x)$, by Lemma 4.1, $M''(v_*) > 0$. This implies that for any v_k sufficiently close to v_* and $v_k \neq v_*$, we have $v_* < M(v_k) < v_k$. By induction, the sequence $\{v_k\}$ monotonically decreases to v_*. The case $v_* > v_r(x)$ can be proven similarly. Finally, notice that when $v_* = v_r(x)$, $M''(v_*) = 0$. In this case, the local behavior of the algorithm around v_* depends on $M'''(v_*)$, which can be either nonnegative or nonpositive. □

Proof of Theorem 4.5:

If v_k converges to v_*, by equation (61) we have

$$\lim_{k \to \infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|^2} = \frac{1}{2} \left| M''(v_*) \right| < \infty.$$

Hence, the `SOR-TS` algorithm with $\omega = 1$ has at least a quadratic order of convergence. In particular, when $v_* = v_r(x)$, the convergence has at least a cubic order. □

Proof of Theorem 6.1:
The `SOR-B` algorithm with $\omega_k \equiv 0$ is always well-defined since for any $v \in \mathbb{R}_+$, $G^B(v)$ is well-defined.

We now show that the `SOR-B` algorithm always converges. First consider the case $x = 0$. Notice $G^B(v; 0, v_*, 0) \equiv \sqrt{2\pi} c_* = v_*$ for $\forall v \in \mathbb{R}_+$. Thus the `SOR-B` algorithm lands on v_* after one iteration.

Now let $x < 0$. Define two function $f(v)$ and $g(v)$ as follows:

$$f(v) = |x| \left[N\Big(\frac{x}{v}\Big) - N\Big(\frac{x}{v_*}\Big) \right] - v_* \left[n\Big(\frac{x}{v}\Big) - n\Big(\frac{x}{v_*}\Big) \right],$$
$$g(v) = |x| \left[N\Big(\frac{x}{v}\Big) - N\Big(\frac{x}{v_*}\Big) \right] - \left[vn\Big(\frac{x}{v}\Big) - v_* n\Big(\frac{x}{v_*}\Big) \right].$$

Then $f(v_*) = g(v_*) = 0$, and

$$f'(v) = \frac{|x|^2 n(x/v)}{v^3}(v - v_*), \quad g'(v) = -n(x/v) < 0.$$

Thus, $f(v) \geq 0$ with equality if and only if $v = v_*$, and $g(v) < 0$ if $v > v_*$. Since $G^B(v) > v_*$ if and only if $f(v) > 0$ when $\omega \equiv 0$, we have $v_1 > v_*$ for any $v_0 \in \mathbb{R}_+$. Since for $v > v_*$, $G^B(v) < v$ if and only if $g(v) < 0$, we have $v_* < v_{k+1} < v_k$ for all $k \in \mathbb{N}$. Thus the sequence $\{v_k\}$ $(k \geq 1)$ monotonically decreases to a limit point v_∞. Taking a limit on the iteration equation then gives $c_* = c^B(x, v_\infty)$. Since vega is strictly positive in the Bachelier formula, we have $v_* = v_\infty$.

To see that the convergence is of quadratic order, notice that when v_k is sufficiently close to v_*, a Taylor expansion of the iteration equation gives

$$v_{k+1} - v_* = \frac{x^2}{2v_*^3}(v_k - v_*)^2 + \mathcal{O}(v_k - v_*)^3,$$

and thus

$$\lim_{k \to +\infty} \frac{|v_{k+1} - v_*|}{|v_k - v_*|^2} = \frac{x^2}{2v_*^3} > 0.$$

□

Proof of Theorem 6.2:
Since $\omega_k = 0$ and $\kappa(x, v) > 0$ for all $x \in \mathbb{R}$ and $v > 0$, $G^{CS}(x; v, x_*, 0)$ is well-defined for any $x \in \mathbb{R}$. Thus, the sequence $\{x_k\}$ is well-defined.

We now show the global convergence of the `SOR-CS` algorithm. We will simply write $G(x)$ for $G^{CS}(x; v, x_*, 0)$. Define two functions $f(x)$ and $g(x)$ by

$$f(x) = e^{x_*} N\Big(\frac{x_*}{v} + \frac{v}{2}\Big) - e^{x_*} N\Big(\frac{x}{v} + \frac{v}{2}\Big) - N\Big(\frac{x_*}{v} - \frac{v}{2}\Big) + N\Big(\frac{x}{v} - \frac{v}{2}\Big),$$
$$g(x) = e^{x_*} N\Big(\frac{x_*}{v} + \frac{v}{2}\Big) - e^{x} N\Big(\frac{x}{v} + \frac{v}{2}\Big) - N\Big(\frac{x_*}{v} - \frac{v}{2}\Big) + N\Big(\frac{x}{v} - \frac{v}{2}\Big).$$

By the definition of $G(x) = G(x; v, x_*, \omega)$, we have

$$f(x) = \left(e^{G(x)} - e^{x_*}\right) N\left(\frac{x}{v} + \frac{v}{2}\right), \quad g(x) = \left(e^{G(x)} - e^{x}\right) N\left(\frac{x}{v} + \frac{v}{2}\right).$$

To examine the signs of $f(x)$ and $g(x)$, notice $f(x_*) = g(x_*) = 0$, and

$$f'(x) = \frac{1}{v} n\left(\frac{x}{v} + \frac{v}{2}\right)\left(e^{x} - e^{x_*}\right), \quad g'(x) = -e^{x} N\left(\frac{x}{v} + \frac{v}{2}\right),$$

where we have used the identity $n\left(\frac{x}{v} + \frac{v}{2}\right)e^{x} = n\left(\frac{x}{v} - \frac{v}{2}\right)$ in deriving the above derivatives. Since $f'(x) > 0$ when $x > x_*$, and $f'(x) < 0$ when $x < x_*$, $f(x)$ achieves minimum at $x = x_*$. Thus $f(x) \geq 0$ on $\mathbb{R}$ with $f(x) = 0$ if and only if $x = x_*$. On the other hand, the function $g(x)$ is monotonically decreasing on $\mathbb{R}$. In particular, for any $x > x_*$, we have $f(x) > 0$ and $g(x) < 0$.

Now let $x_0 \neq x_*$ in the `SOR-CS` algorithm. We have $f(x_0) > 0$ by the above analysis. That is, $x_1 = G(x) > x_*$. The above analysis then shows that $x_* < x_{k+1} < x_k$ for any $k \in \mathbb{N}$. Thus x_k monotonically decreases to a limit x_∞. It can be easily shown that $x_\infty = x_*$ since x_* is the only fixed point of $G(x)$.

To see the convergence is of quadratic order, notice that when x_k is sufficiently close to x_*, a Taylor expansion of the iteration equation gives

$$x_{k+1} - x_* = \frac{n\left(\frac{x}{v} + \frac{v}{2}\right)}{2vN\left(\frac{x}{v} + \frac{v}{2}\right)}(x_k - x_*)^2 + \mathcal{O}(x_k - x_*)^3,$$

and thus

$$\lim_{k \to +\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^2} = \frac{n\left(\frac{x}{v} + \frac{v}{2}\right)}{2vN\left(\frac{x}{v} + \frac{v}{2}\right)} > 0.$$

□

References

Bachelier, L. 1900. Théorie de la spéculation, *Annales de l'Ecole Normale Superiure* **17** 21–86.

Bharadia, M. A., N. Christofides, G. R. Salkin. 1996. Computing the Black Scholes implied volatility. *Advances in Futures and Options Research* **8** 15–29.

Black, F., M. Scholes. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy* **81** 637–654.

Brenner, M., M. G. Subrahmanyam. 1988. A simple formula to compute the implied standard deviation. *Financial Analysts Journal* **5** 80–83.

Carr, P., L. Wu. 2003. The finite moment log stable process and option pricing. *The Journal of Finance* **58** 753–777.

Chambers, D. R., S. J. Nawalkha. 2001. An improved approach to computing implied volatility. *The Financial Review* **38** 89–100.

Chance, D. M. 1993. Leap into the unknown. *Risk* **6** 60–66.

Chance, D. M. 1996. A generalized simple formula to compute the implied volatility. *The Financial Review* **31** 859–867.

Chandrasekhar, C. R., R. Gukhal. 2004. The compound option approach to American options on jump-diffusions. *Journal of Economic Dynamics and Control* **28** 2055–2074.

Chargoy-Corona, J., C. Ibarra-Valdez. 2006. A note on Black-Scholes implied volatility. *Physica A* **370** 681–688.

Choi, J., K. Kim, M. Kwak. 2007. Numerical approximation of the implied volatility under arithmetic Brownian motion. Available at SSRN: `http://ssrn.com/abstract=990747.`

Corrado, C. J., T. W. Jr Miller. 1996a. A note on a simple, accurate formula to compute implied standard deviations. *Journal of Banking and Finance* **20** 595–603.

Corrado, C. J., T. W. Jr Miller. 1996b. Volatility without tears. *Risk* **7** 49–52.

Fischer, S. 1978. Call option pricing when the exercise price is uncertain, and the valuation of index bonds. *The Journal of Finance* **33** 169–176.

Geske, R. 1977. The valuation of corporate liabilities as compound options. *Journal of Financial and Quantitative Analysis* **12** 541–552.

Geske, R. 1979. The valuation of compound options. *Journal of Financial Economics* **7** 63–81.

Heston, S. L. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies* **6** 327–343.

Hodges, S. D., M. J. P. Selby. 1987. On the evaluation of compound options. *Management Science* **33(3)** 347–355.

Isaacson, E., H. B. Keller. 1994. *Analysis of Numerical Methods*. Dover Publications, New York.

Jäckel, P. 2006. By implication. *Wilmott* (November 2006) 60–66.

Kelly, M. A. 2006. Faster implied volatilities via the implicit function theorem. *The Financial Review* **41(4)** 589–597.

Latané, H., R. Rendleman. 1976. Standard deviations of stock price ratios implied in option prices. *The Journal of Finance* **31** 369–382.

Li, M. 2007. The impact of return nonnormality on exchange options. *Journal of Futures Markets* (forthcoming).

Li, M. 2008. Approximate inversion of the Black-Scholes formula using rational functions. *European Journal of Operations Research* **185(2)** 743–759.

Manaster, S., G. Koehler. 1982. The calculation of implied variances from the Black-Scholes model. *The Journal of Finance* **38** 227–230.

Margrabe, W. 1978. The value of an option to exchange one asset for another. *The Journal of Finance* **33** 177–186.

Merton, R. C. 1973. The theory of rational option pricing. *Bell Journal of Economics and Management Science* **4** 141–183.

Merton, R. C. 1976. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics* **3** 125–143.

Musiela, M., M. Rutkowski. 2005. *Martingale Methods In Financial Modelling (2nd edition)*, Springer, New York.

Nash, S. G. 1990. *A History of Scientific Computing.* ACM Press, New York, 180–194.

Nelder, J. A, R. Mead. 1965. A simplex method for function minimization. *Computer Journal* **7** 308–313.

Press, W. H., B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. 1992. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge.

Schachermayer, W., J. Teichmann. 2007. How close are the option pricing formulas of Bachelier and Black-Merton-Scholes? *Mathematical Finance* (forthcoming).

Schmalensee, R., R. Trippi. 1978. Common stock volatility expectations implied by option premia. *The Journal of Finance* **33** 129–147.

Sidi, A. 2002. *Practical Extrapolation Methods: Theory and Applications.* Cambridge University Press, Cambridge.

Stoll, H. R. 1969. The relationship between put and call option prices. *The Journal of Finance* **31** 319–332.

Young, D. M. 1954. Iterative methods for solving partial difference equations of elliptic type. *Transactions of the Mathematical Society* **76** 92–111.

Table 1
The SOR, SOR-DR and SOR-TS algorithms

This table gives the values of v_k for the first five iterations in the `SOR`, `SOR-DR` and `SOR-TS` algorithms. The option used has moneyness $x = -0.5$ and volatility $v_* = 1$ so that $\Phi(v_*) = 0$. For the `SOR` and `SOR-TS` algorithms, we set $\omega = 1$ for all the iterations. The initial estimate is chosen to be $v_0 = 0.6$ or 1.4 for all three algorithms. All the rows are computed with (10^{-16})-precision arithmetic, except that the rows $4'$ and $5'$ are computed with (10^{-50})-precision arithmetic.

Panel A: $v_0 < v_*$

	SOR		SOR-DR			SOR-TS		
k	v_k	$\lvert v_k - v_*\rvert$	v_k	$\lvert v_k - v_*\rvert$	ω_k	v_k	$\lvert v_k - v_*\rvert$	α_k
0	0.6000	0.4000	0.6000	0.4000	−0.4706	0.6000	0.4000	3.7778
1	0.7284	0.2716	1.2429	0.2429	0.2141	1.0850	0.0850	1.8495
2	0.8327	0.1673	1.0192	0.0192	0.0190	1.0016	0.0016	1.9968
3	0.9050	0.0950	1.0002	2×10^{-4}	2×10^{-4}	1.0000	6×10^{-7}	2.0000
4	0.9489	0.0511	1.0000	2×10^{-8}	2×10^{-8}	1.0000	1×10^{-13}	2.0000
5	0.9735	0.0265	1.0000	7×10^{-16}		1.0000	4×10^{-16}	
$5'$	0.9735	0.0265	1.0000	1×10^{-16}		1.0000	3×10^{-27}	

Panel B: $v_0 > v_*$

	SOR		SOR-DR			SOR-TS		
k	v_k	$\lvert v_k - v_*\rvert$	v_k	$\lvert v_k - v_*\rvert$	ω_k	v_k	$\lvert v_k - v_*\rvert$	α_k
0	1.4000	0.4000	1.4000	0.4000	0.3243	1.4000	0.4000	1.5102
1	1.1507	0.1507	1.0413	0.0413	0.0404	1.0235	0.0235	1.9546
2	1.0675	0.0675	1.0008	8×10^{-4}	8×10^{-4}	1.0001	1×10^{-4}	1.9997
3	1.0321	0.0321	1.0000	3×10^{-7}	3×10^{-7}	1.0000	4×10^{-9}	2.0000
4	1.0157	0.0157	1.0000	5×10^{-14}	5×10^{-14}	1.0000	4×10^{-16}	2.0000
5	1.0077	0.0077	1.0000	1×10^{-16}		1.0000	4×10^{-16}	
$4'$	1.0157	0.0157	1.0000	5×10^{-14}	5×10^{-14}	1.0000	5×10^{-18}	2.0000
$5'$	1.0077	0.0077	1.0000	1×10^{-27}		1.0000	6×10^{-36}	

Table 2
Accuracy of the `SOR`, `SOR-DR` and `SOR-TS` algorithms in D^-

This table gives the accuracy of the `SOR` , `SOR-DR` and `SOR-TS` algorithms in domain D^-. All three algorithms are implemented with (10^{-16})-precision arithmetic in MATLAB 7.1 on a Dell Dimension 4600 desktop computer (2.8 GHz, 1G RAM). The relaxation parameter ω is chosen to be 1 in both `SOR` and `SOR-TS` algorithms. The result for $k = 0$ represents the rational approximation, which is the same for all three algorithms. The means, medians and maximums are calculated by uniformly and densely populating the domain D^- with roughly 1 million options. An asterisk (*) indicates that the number is in the order of or smaller than the machine accuracy (10^{-16}). For a given k, the computing times reported are for all the options in the domain D^-.

	`SOR`			`SOR-DR`			`SOR-TS`		
	mean	median	max	mean	median	max	mean	median	max
$\lvert v_0 - v_*\rvert$	0.356	0.319	0.783						
$\lvert c_0 - c_*\rvert$	0.038	0.022	0.290						
Time (s)		0.65							
$\lvert v_4 - v_*\rvert$	0.003	2×10^{-6}	0.078	2×10^{-11}	1×10^{-15}	2×10^{-8}	5×10^{-11}	$*$	2×10^{-8}
$\lvert c_4 - c_*\rvert$	9×10^{-4}	2×10^{-7}	0.024	6×10^{-12}	$*$	6×10^{-9}	4×10^{-12}	$*$	5×10^{-10}
Time (s)		9.43			9.70			10.43	
$\lvert v_5 - v_*\rvert$	0.002	8×10^{-8}	0.058	4×10^{-15}	$*$	1×10^{-13}	3×10^{-15}	$*$	1×10^{-13}
$\lvert c_5 - c_*\rvert$	6×10^{-4}	8×10^{-9}	0.015	$*$	$*$	4×10^{-14}	$*$	$*$	2×10^{-14}
Time (s)		11.58			12.60			13.17	

Table 3
Comparing the Newton-Raphson, Dekker-Brent, and SOR-TS algorithms

This table compares the performance of the Newton-Raphson (`NR`) algorithm, the Dekker-Brent (`DB`) algorithm and the `SOR-TS` algorithm for a particular option (x, v_*), where we let $x = -1$ and $v_* = 2$. We start all three algorithms from three different initial values, namely, 0.1, 4, and 20. The Newton-Raphson algorithm is implemented naively, that is, without any safeguarding feature such as checking for positivity. The initial Dekker-Brent bracketed region is taken to be $[0.0005, \max(v_0, 10)]$. Panel A reports the values of each iteration while Panel B reports the errors. `Inf` is the IEEE arithmetic representation for positive infinity, while `NaN` is the IEEE arithmetic representation for Not-a-Number. A `NaN` is obtained as a result of mathematically undefined operations like 0.0/0.0 and `Inf`−`Inf`. An asterisk (*) indicates that the number is in the order of or smaller than the machine accuracy (10^{-16}).

Panel A: v_k

k	NR			DB			SOR-TS		
0	0.1	4	20	0.1	4	20	0.1	4	20
1	4×10^{21}	−0.8315	-4×10^{21}	0.0005	0.0005	0.0005	161.14	2.0292	2.1750
2	Inf	7.0268	Inf	10	2.2010	10.1975	2.2515	2.0000	2.0011
3	NaN	−353.15	NaN	5.1476	1.8917	5.0990	2.0022	2.0000	2.0000
4	NaN	Inf	NaN	2.6238	2.0042	2.5497	2.0000	2.0000	2.0000
5	NaN	Inf	NaN	2.0668	2.0001	2.0518	2.0000	2.0000	2.0000

Panel B: $|v_k - v_*|$

k	NR			DB			SOR-TS		
0	1.9	2	18	1.9	2	18	1.9	2	18
1	4×10^{21}	2.8315	-4×10^{21}	1.9995	1.9995	1.9995	159.14	0.0292	0.1750
2	Inf	5.0268	Inf	8	0.2010	8.1975	0.2515	3×10^{-5}	0.0011
3	NaN	355.15	NaN	3.1476	0.1083	3.0990	0.0022	7×10^{-11}	7×10^{-8}
4	NaN	Inf	NaN	0.6238	0.0042	0.5497	2×10^{-7}	*	*
5	NaN	Inf	NaN	0.0668	0.0001	0.0518	3×10^{-15}	*	*

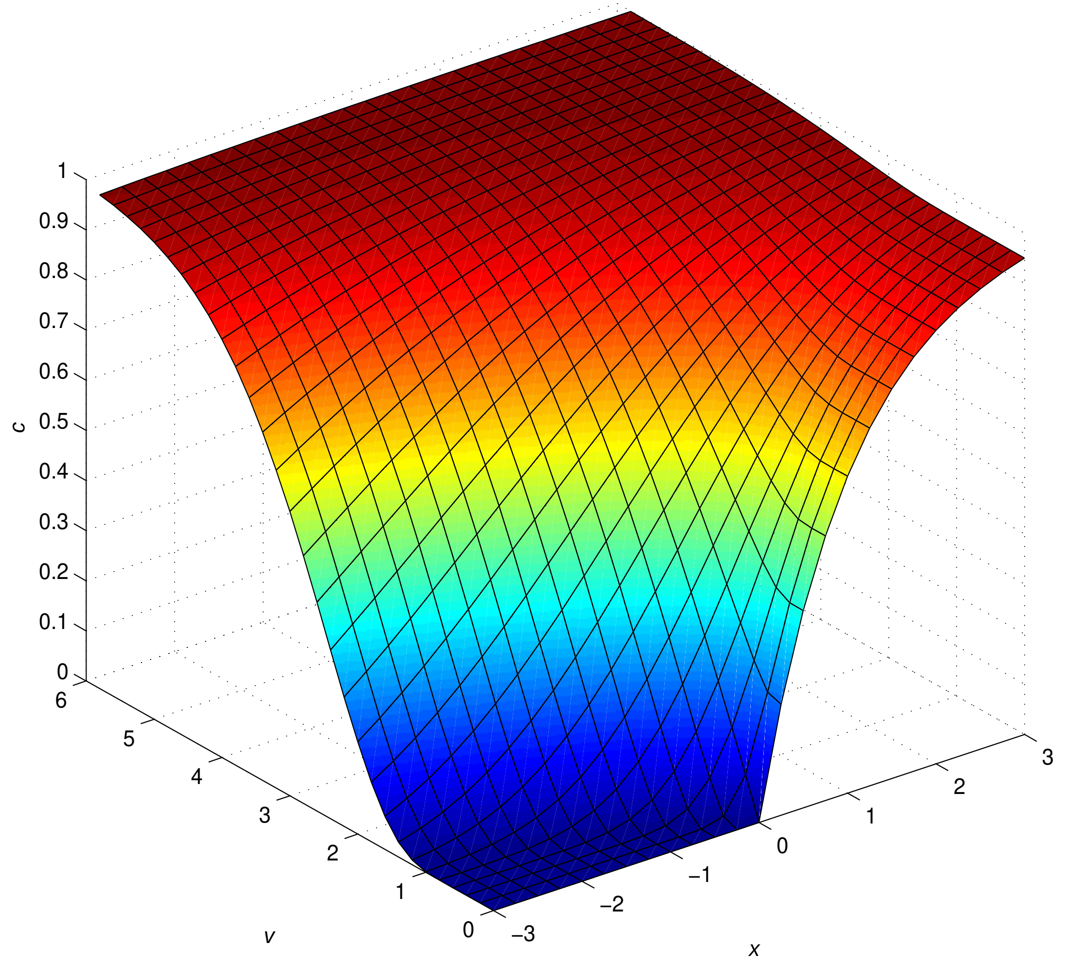

Figure 1. Normalized Black-Scholes call price c as a function of moneyness x and total volatility v. This surface implicitly gives v as a function of x and c. In the regions where $x/v \leq -3$ and $x/v \geq 3$, option prices are very insensitive to changes in v, implying that the inversion of Black-Scholes formula is not very meaningful in these regions.

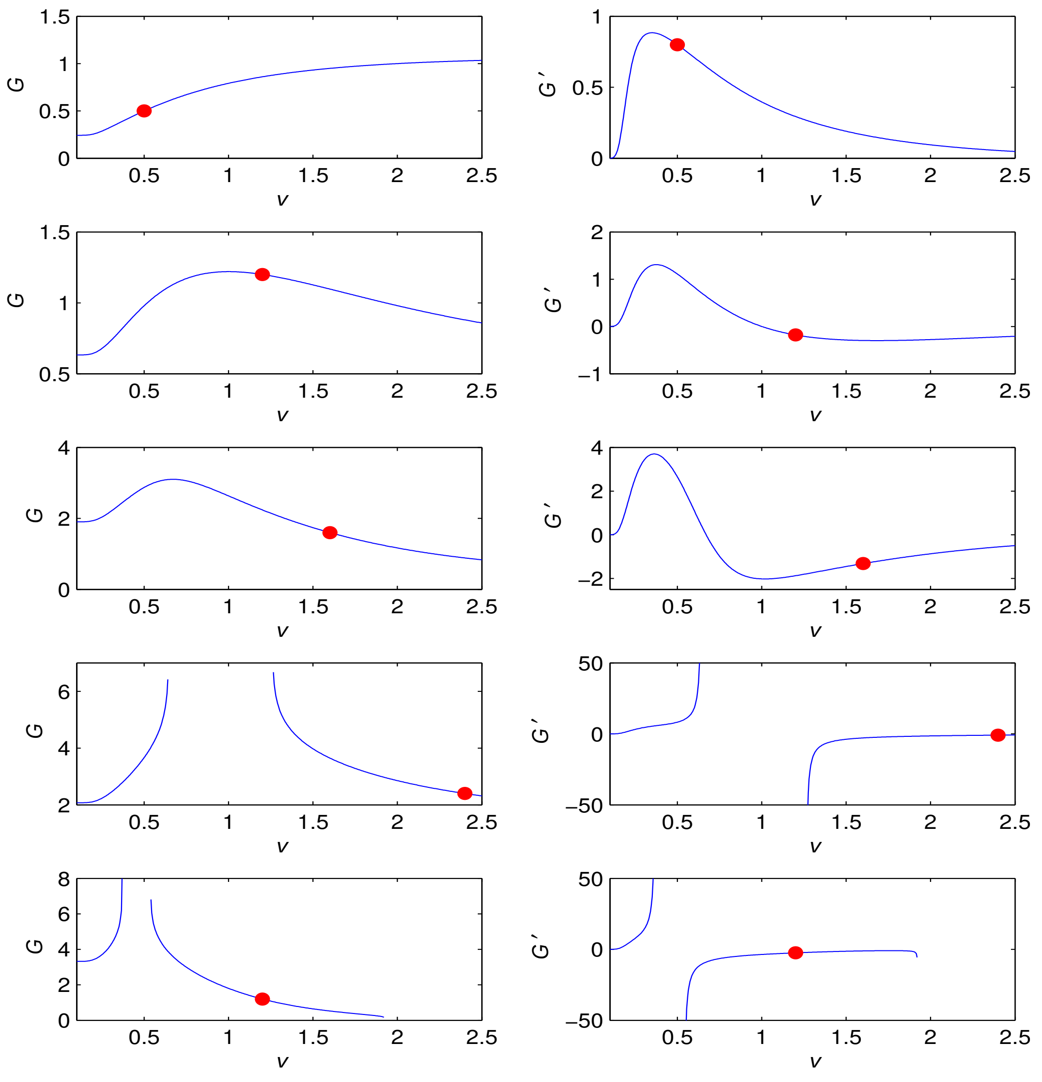

Figure 2. The function $G(v; x, v_*, \omega)$ and its derivative $G'(v; x, v_*, \omega)$. Each row corresponds to a different combination of parameters (x, v_*, ω). The dots indicate the positions of v_*. As the graphs show, the function $G(v)$ can fail to be globally well-defined, globally contracting, or locally contracting.

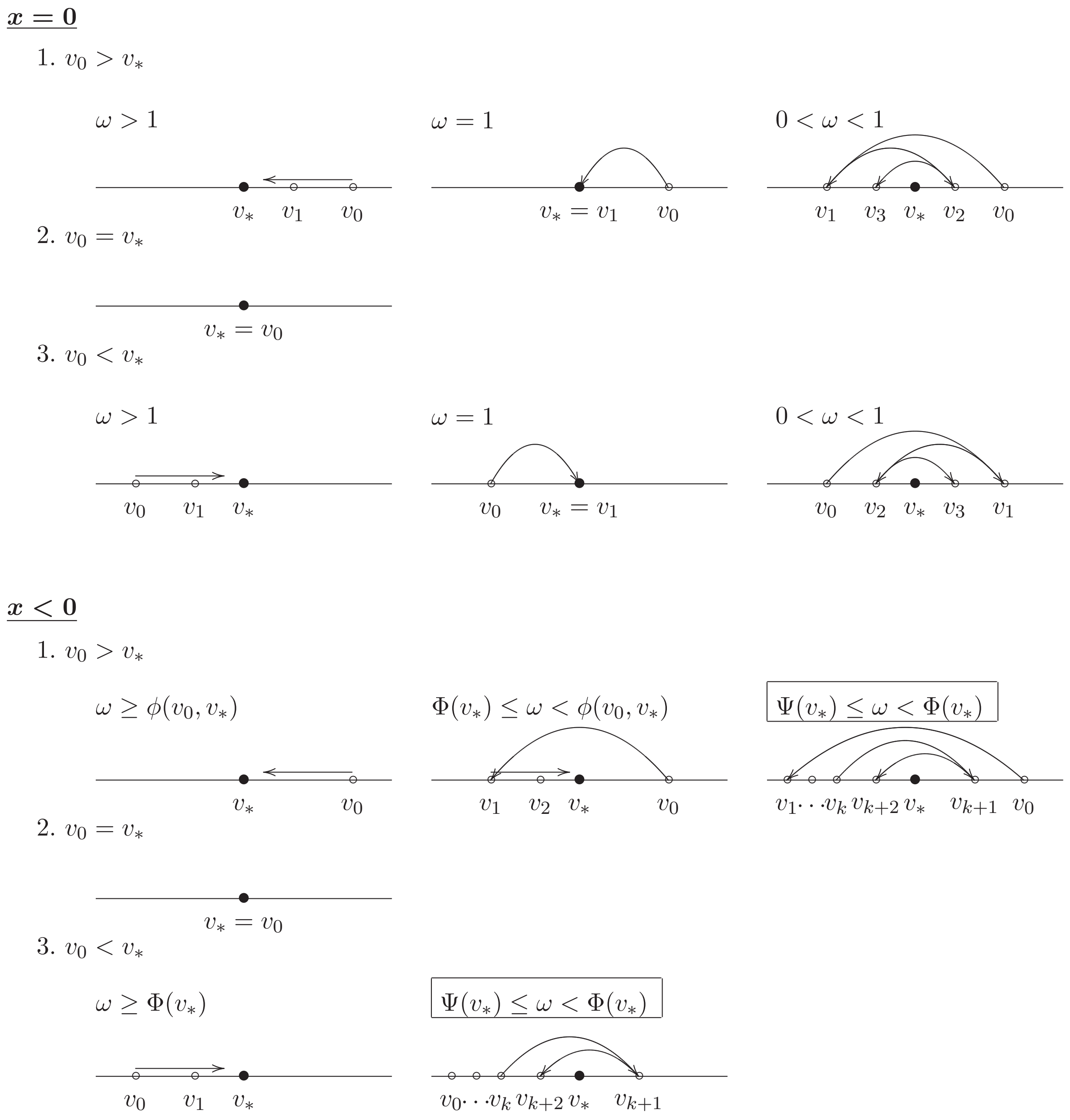

Figure 3. Convergence patterns for the `SOR` algorithm. This figure shows all the possible convergence patterns for the `SOR` algorithm. The two cases $x = 0$ and $x < 0$ are listed separately. Except for the two cases whose conditions are boxed, all the conditions above each subplots are both sufficient and necessary for each particular convergence pattern to occur. The two boxed conditions are only necessary conditions.

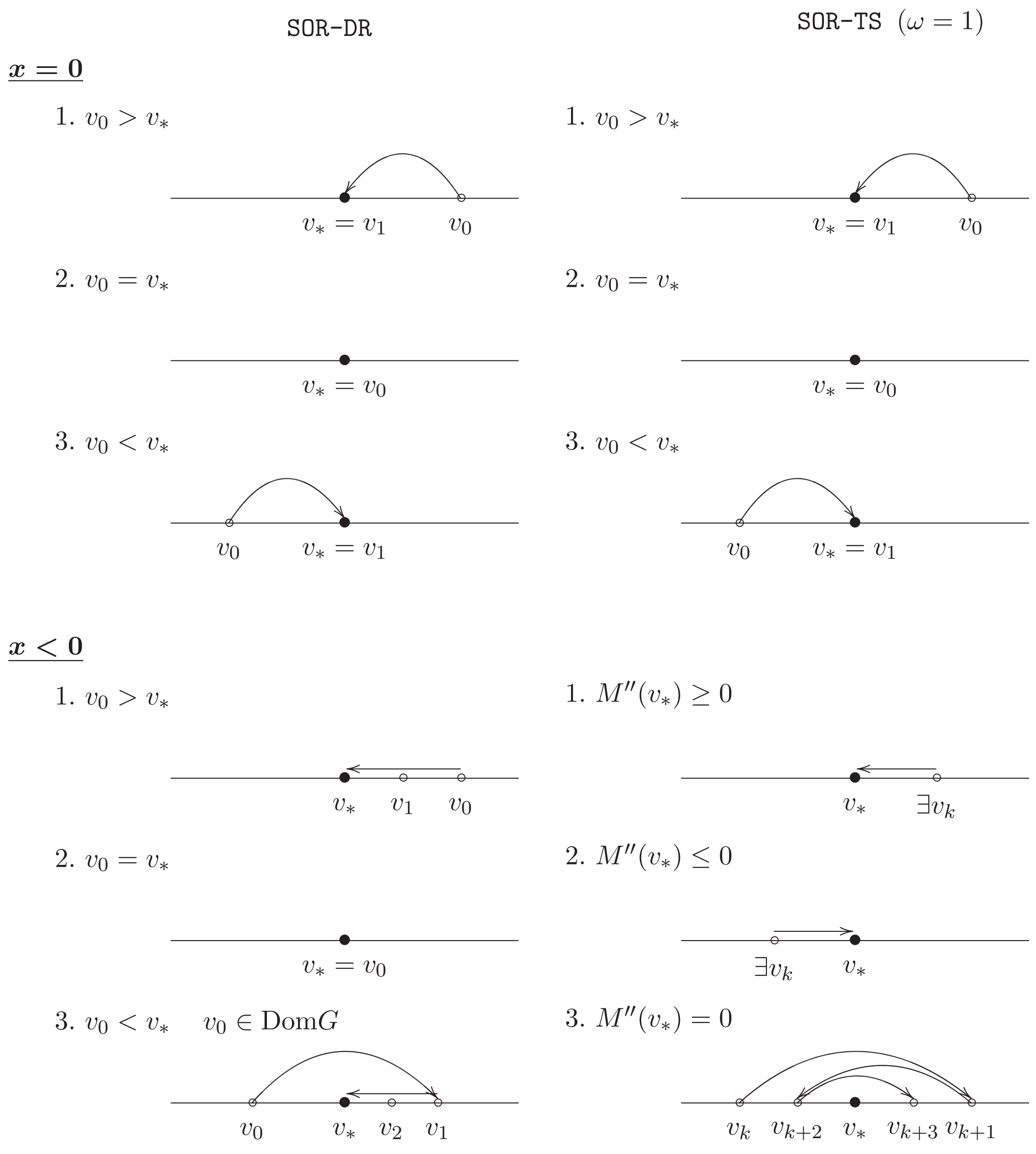

Figure 4. Convergence patterns for the SOR-DR and SOR-TS $(\omega = 1)$ algorithms. The left and right half of the figure shows all the possible convergence patterns for the SOR-DR and SOR-TS $\omega = 1$ algorithms, respectively. The two cases $x = 0$ and $x < 0$ are listed separately. The two algorithms are always globally well-defined, except that when $v_0 < v_*$ and $x < 0$, we need $G(v_0)$ to be well-defined to guarantee that $\{v_k\}$ is defined.

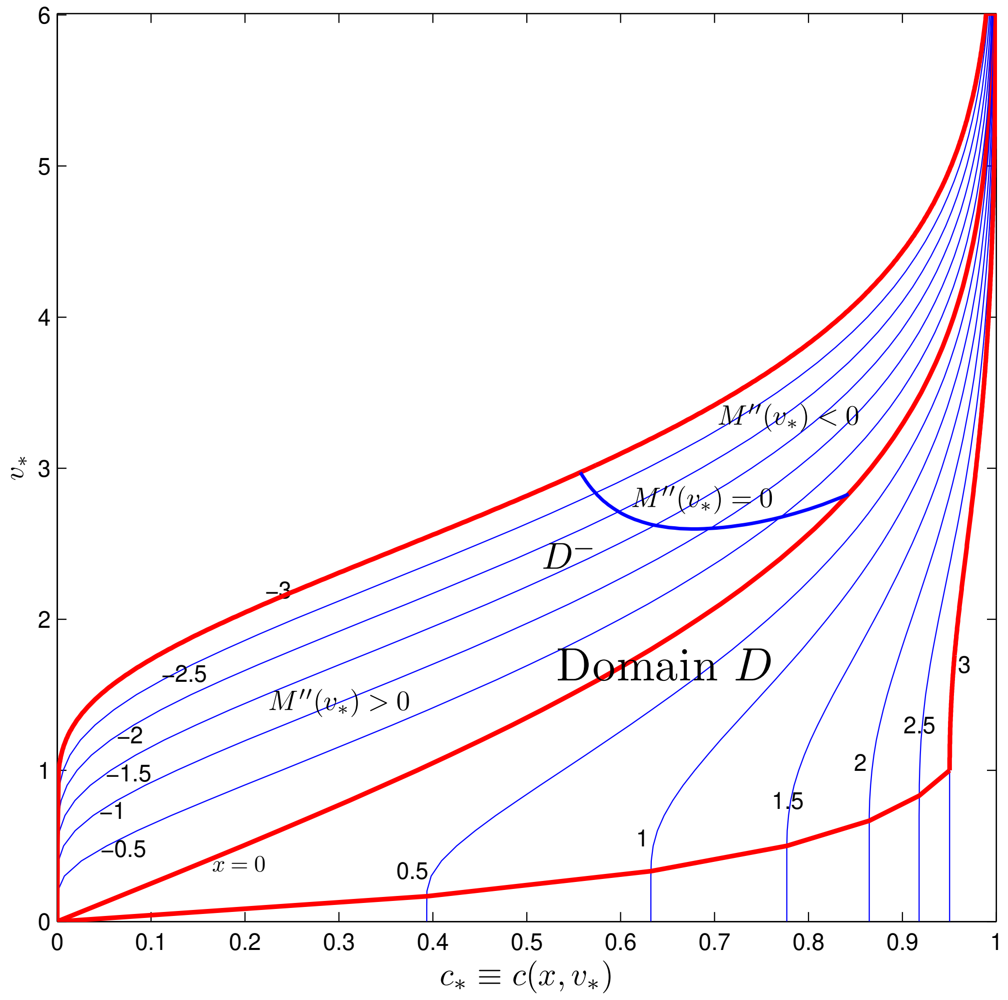

Figure 5. Domain of inversion D**.** D is constructed using four criteria, namely, $0.0005 \leq v_* \leq 6$, $|x| \leq 3$, $0.0005 \leq c_* \leq 0.9995$ and $|x|/v_* \leq 3$. Because of the “in-and-out” duality, only options with $x \leq 0$ will be considered. The curve $M''(v_*) = 0$ divides the left half domain D^- further into two parts.